1. (21pts.) BASICS

Perform the following 8-bit number conversions as requested:

\[ \begin{align*}
\text{\$8B} & = \text{11110001}_{10} \text{ (signed)} \\
\%10111110 & = \text{276}_8 \text{ (unsigned)} \\
183 & = \text{\$B7} \\
68 & = \%01000000 \\
-98 & = \text{\$9E}
\end{align*} \]

For the following questions, assume \((C)=1\) before each code sequence.

What are the values of \(B\): \(\text{\$DA} \quad \text{N: 1} \quad \text{Z: } \phi \quad \text{V: } \phi \quad \text{C: 1}\)
following execution of the code segment below:

\[
\begin{align*}
\text{LDAB} & \quad \#137 \\
\text{SUBB} & \quad \#\$AF
\end{align*}
\]

What are the values of \(A\): \(\text{\$B5} \quad \text{N: 1} \quad \text{Z: } \phi \quad \text{C: 1}\)
following execution of the code segment below:

\[
\begin{align*}
\text{LDAA} & \quad \#\$76 \\
\text{ROLA} \\
\text{ROLA} \\
\text{ROLA}
\end{align*}
\]

What are the values of \(B\): \(\text{\$B5} \quad \text{N: 1} \quad \text{Z: } \phi \quad \text{V: } -(0)\)
following execution of the code segment below:

\[
\begin{align*}
\text{LDAB} & \quad \#127 \\
\text{ANDB} & \quad \%11001100 \\
\text{EORB} & \quad \#\$F0
\end{align*}
\]
2. (25 pts.) ANALYSIS: The following code sequence is executed with initial values of memory and registers as shown (ALL numbers are hexadecimal). Fill in the table with the results of each instruction execution. Show only changes, and show in hexadecimal without the $ symbol. This is demonstrated with the execution of the first instruction: $A000

<table>
<thead>
<tr>
<th>Instruction Sequence</th>
<th>A</th>
<th>B</th>
<th>X</th>
<th>Y</th>
<th>SP</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
<th>Addressing Mode</th>
<th>Effective Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldxy $A000</td>
<td>F0</td>
<td>0F</td>
<td>A003</td>
<td>FFFF</td>
<td>E800</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>EXT</td>
<td>A000</td>
</tr>
<tr>
<td>ldab #0</td>
<td>00</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>IMM</td>
<td>-</td>
</tr>
<tr>
<td>pshb</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INH</td>
<td>-</td>
</tr>
<tr>
<td>adda 2,x</td>
<td>E4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>INDEXED</td>
<td>A005</td>
</tr>
<tr>
<td>oraa 1,x</td>
<td>E4</td>
<td></td>
<td></td>
<td>A002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BRESF INDEXED</td>
<td>A003</td>
</tr>
<tr>
<td>leas -4,SP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IDX</td>
<td>A0 FEF</td>
</tr>
<tr>
<td>rora</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INH</td>
<td>-1 ea</td>
</tr>
</tbody>
</table>

-1 ea
3. (24 pts.) **UNDERSTANDING CODE:** Lucky would like for you to review her code but doesn’t have time to explain it to you. Here it is:

```assembly
; This program does computations using array Vec.
; The results are stored in Res.
; Owner: Lucky  Last changed: 9-25-2008  Status: Excellent comments!

N EQU 24 ;Array Count
Key EQU 1024 ;Key

ORG $800 ;Start data area at $800
Vec DS.W 32 ;Reserve Vec
Res DS.B 1 ;Reserve a byte for Res

ORG $1000 ;Start program at $1000
LDY #N ;Set up loop count in Y register
CLRA ;Set A register to $00
STAA Res ;Store A register in Res
LDD #Key ;Load D register with Key
LDX #Vec ;Load X with Vec
Loop CPD 0,X ;Compare D with array element
INX ;increment pointer x (?)
INX ;increment pointer x (?)
BBQ Found ;If compare in “loop” matches, Go to Found
DBNE Y,Loop ;Have we finished?
RTS ;Done, so return.
Found INC Res ;Increment Result
BRA Loop ;Go back to loop
END ;Done!
```

a. What is this routine trying to do? Be concise.

Put the # of occurrences of word value 1024 found in Vec into Res.

b. There are 2 known coding errors in this module. What are they? (Don’t guess)

BEQ DOESN’T TEST COMPARE. IT TESTS INX Result $x $0 (never)
-lead
RTS WRONG. SHOULD BE SWI

---

c. What values does the assembler/loader place in the following memory addresses?

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>($1000)</td>
<td>$CD</td>
</tr>
<tr>
<td>($1001)</td>
<td>$FF</td>
</tr>
<tr>
<td>($1002)</td>
<td>$18</td>
</tr>
<tr>
<td>($1003)</td>
<td>$87</td>
</tr>
<tr>
<td>($1004)</td>
<td>$7A</td>
</tr>
<tr>
<td>($1005)</td>
<td>$08</td>
</tr>
</tbody>
</table>

d. In line 6, what is smallest value of N that would work? ___ the largest? 32 ___

e. Lines 19-21 can be simplified. Show exactly how:

19 Loop CPD 2, X+ ; Also fixes first coding error

f. Lucky thinks this code is well documented. What is your professional opinion?

No. It simply document semantics (behavior) of instruction set.
4. **UNDERSTANDING STACKS AND SUBROUTINES**: Consider the following test program and subroutine "Smart" below:

```assembly
; Main Program
; Data Area
ORG $1000
Array1 DS.B $100
Varl DC.W $ABCD
Stktop EQU $BFFF ;Highest mem reserved for Stack
; Program Area
; Test Program to exercise subroutine Smart
ORG $2000
LDI #Stktop+1 ; Initialize top of stack
LDD Varl ; Pass value of first parameter
PSHD
LDX #Array1 ; Pass next parameter
PISHX
JSR Smart ; Call subroutine "Smart" with Varl and Array
Done SWI ; Done. Return to Monitor
; Subroutine Smart
; Save Registers that Smart will use (B and X)
Smart PSHB
PISH
LEAS -2, SP ; What does this do?
; Smart code here
; Restore registers and stack pointer used by Smart
RTS
END
```

a. On the next (opposing) page, fill in the registers and memory table with hex values where memory values are defined, and show (to left of table) where pointer SP points when the processor reaches the location commented " ; Smart code here". If a value can't be determined, leave it blank. If it is an instruction address, use its label (e.g. Done) rather than computing the address.

In addition, to the right side of the memory map show where (i.e. annotate)
1. incoming parameters,
2. the return address (from call to Smart),
3. saved registers, (from any subroutine calls) and
4. local storage (in Smart)
are stored.

b. What actual code would be used to replace the commented line:
   ; Restore registers and stack pointer used by Smart
   LEAS +2, SP
   PULX
   PULB
   

c. The main program passed Array1 to Smart as a parameter. Show one line of code in Smart that would put this parameter into the X register:
   LDX, 7, SP
4. (Continued)

d. In the main program how is parameter Var passed (value or reference)? **value**
   How is Array passed? **reference**

Memory map for question 4a:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP</td>
<td>BFF5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFF4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFF5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFF6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFF7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFF8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFF9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFFA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFFB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFFC</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFFD</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFFE</td>
<td>AB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$BFFF</td>
<td>CD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

{ Local Storage
{ Saved Registers
{ Return Address
{ Incoming Parameters

5. (7 Pts.) I/O You want to create an IRQ interrupt when you push a button. Draw a simple circuit to achieve this, using the IRQ* pin of the MC68HC12 shown below.

![Circuit Diagram]

Describe the three key things you must do (besides your circuit and writing the ISR for this interrupt) to make the processor respond correctly to an interrupt from your pushbutton:

1. **Put address of ISR in interrupt vector**
2. **Enable IRQ interrupts (IRQ I/O register)**
3. **Enable maskable interrupts (CLI)**