Solutions – Midterm Exam
(October 21st @ 5:30 pm)
Clarity is very important! Show your procedure!

**Problem 1 (20 pts)**

1. **(5 pts)** Complete the following table. The numbers are unsigned integers.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>BCD (bits)</th>
<th>Binary</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>181</td>
<td>0001 1000 0001</td>
<td>10110101</td>
<td>B5</td>
</tr>
<tr>
<td>59</td>
<td>01011001</td>
<td>00111111</td>
<td>3B</td>
</tr>
<tr>
<td>86</td>
<td>10000110</td>
<td>01010110</td>
<td>56</td>
</tr>
<tr>
<td>114</td>
<td>0001 0001 0100</td>
<td>01100100</td>
<td>72</td>
</tr>
</tbody>
</table>

2. **(5 pts)** Complete the following table. The numbers are represented with 8 bits.

<table>
<thead>
<tr>
<th>Representation</th>
<th>Decimal</th>
<th>1's complement</th>
<th>2's complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>-50</td>
<td>-10001101</td>
<td>-10001110</td>
<td></td>
</tr>
<tr>
<td>-109</td>
<td>10010010</td>
<td>10010011</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>01001101</td>
<td>01001101</td>
<td></td>
</tr>
<tr>
<td>-86</td>
<td>10101010</td>
<td>10101010</td>
<td></td>
</tr>
</tbody>
</table>

3. **(5 pts)** Perform the following addition and subtraction of 8-bit unsigned integers. Indicate (every carry) or borrow from $c_8$ to $c_0$ (or $b_0$ to $b_8$). For the addition, determine whether there is an overflow. For the subtraction, determine whether we need to keep borrowing from a higher byte.

**Example:**

- **54 + 210**

\[
\begin{array}{c}
54 = 0x36 = \overline{0011 0110} + \\
210 = 0xD2 = \overline{1101 0010} \\
\hline
\overline{1001 0000} = 0x44
\end{array}
\]

Overflow! → 1 0 0 0 0 0 1 0 0 0

- **77 - 194**

\[
\begin{array}{c}
77 = 0x4D = \overline{0100 1101} - \\
194 = 0xC2 = \overline{1100 0010} \\
\hline
\overline{0000 0011}
\end{array}
\]

Overflow! → 1 0 0 0 0 0 1 0 1 1

- **86 + 181**

\[
\begin{array}{c}
86 = 0x56 = \overline{0101 0110} + \\
181 = 0xB5 = \overline{1011 0101} \\
\hline
\overline{1000 0011}
\end{array}
\]

Overflow! → 1 0 0 0 0 1 0 1 1

- **86 - 181**

\[
\begin{array}{c}
86 = 0x56 = \overline{0101 0110} - \\
181 = 0xB5 = \overline{1011 0101} \\
\hline
\overline{1010 0001}
\end{array}
\]

Overflow! → 1 0 1 0 0 0 0 1

4. **(5 pts)** Perform the following operations using the 2's complement representation with 8 bits. Determine whether the operations result in an overflow.

- **-59 - 114**

\[
\begin{array}{c}
-59 = 0xC5 = \overline{1100 0010} + \\
-114 = 0x8E = \overline{1000 1110} \\
\hline
\overline{0x53} = \overline{0101 0011}
\end{array}
\]

Overflow! → 0x53 = 0101 0011

- **-86 + 114**

\[
\begin{array}{c}
-86 = 0xAA = \overline{1010 1010} + \\
114 = 0x72 = \overline{0111 0010} \\
\hline
\overline{28} = \overline{0001 1110}
\end{array}
\]

No Overflow → 28 = 0x1C = 0001 1110

Instructor: Daniel Llamocca
PROBLEM 2 (10 PTS)
- A microprocessor has a 16-bit address line, where each address contains 8 bits. An SRAM device is connected to the microprocessor. The microprocessor has assigned the addresses 0xD800 to 0xDFFF to this SRAM.
  - What is the size (in KB, or MB) of this SRAM?
  - What is the minimum number of bits required to represent the addresses only for this SRAM?

✓ The range 0xD800 to 0xDFFF is akin to all possible cases with 11 bits. Thus the SRAM size is $2^{11}$ bytes = 1 KB.
✓ We only need 11 bits for this SRAM.

PROBLEM 3 (20 PTS)
Given the following set of instructions, complete the following:
- Register values (in hexadecimal format) as the instructions are executed.
- The state of the memory contents (in hexadecimal format) after the last instruction has been executed. Also, specify the memory address at which the contents of $D$ are stored (last instruction).
- The addressing mode of each instruction. Be specific, if for example the addressing mode is indexed, indicate which one in particular. Note that the movw instruction uses two addressing modes.

<table>
<thead>
<tr>
<th>Addressing Mode</th>
<th>clra</th>
<th>clrb</th>
<th>ldx #$FADE</th>
<th>ldy #$1A00</th>
<th>movw #$1E20,1,Y+</th>
<th>ldab #$81</th>
<th>sex b,d</th>
<th>addd 1,-Y</th>
<th>exg x,y</th>
<th>std [0,X]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$FF</td>
<td>$1D</td>
<td>$1D</td>
</tr>
<tr>
<td>Indexed, Indexed - Post-Increment</td>
<td></td>
<td></td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$FF</td>
<td>$1D</td>
<td>$1D</td>
</tr>
<tr>
<td>Immediate</td>
<td></td>
<td></td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$FF</td>
<td>$1D</td>
<td>$1D</td>
</tr>
<tr>
<td>Indexed - Pre-Decrement</td>
<td></td>
<td></td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$FF</td>
<td>$1D</td>
<td>$1D</td>
</tr>
<tr>
<td>Indexed Indirect - 16 bit Offset</td>
<td></td>
<td></td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
<td>$00</td>
</tr>
</tbody>
</table>

Address where $D$ is stored

| Address where $D$ is stored | 0x1A00 | $1E     | 0x1A01 | $20     | ...     | 0x1E20 | $1D     | 0x1E21 | $A1     |
PROBLEM 4 (10 PTS)

- Mark the correct option:
  - ✔️ The Interrupt Vector Table contains the list of: **Vector Addresses** **Interrupt Vectors**
  - ✔️ The Software Interrupt (swi) is a: **Maskable Interrupt** **Non-maskable Interrupt**

- Determine whether the following statements are True or False. If the statement is false, explain why.
  - ✔️ Inside an Interrupt Service Routine, the values of the PC and CPU registers are pushed in the Stack.
    - **FALSE.** The ISR does not do this. The processor does this before the ISR is executed.
  - ✔️ An Interrupt Vector is the starting address of an Interrupt Service Routine.
    - **TRUE**
  - ✔️ When servicing a Reset, the values of the PC and CPU Registers are pushed in the Stack.
    - **FALSE.** The processor does not save these registers, as the Reset will initialize these values.

- Complete:
  - ✔️ To enable/disable all maskable Interrupts, we configure the bit _I_ of CCR.
  - ✔️ The /XIRQ Interrupt is enabled by setting the bit _X_ of CCR to 0.

PROBLEM 5 (20 PTS)

- (5 pts) Complete the Assembly Program below so that the state of bits 5 down to 1 on the DIP Switch is displayed only on the 5 leftmost bits on the LEDs (PORT B). The figure shows an example on the Dragon12-Light Board: the number 10011 is shown on the five leftmost LEDs, while the other LEDs are off.

```assembly
; code section
ORG ROMStart

Entry:
_Startup:
LDS #$4000
movb #$FF, DDRB
movb #$00, DDRH

showDIPSW: ldaa PTH
            anda #$3E
            lsla
            lsla
            staa PORTB
            bra showDIPSW
```

; /* Write instructions here */

; /* End of your instructions */
sta PORTB ; Contents of register A are written on PORTB
bra showDIPSW
(5 pts) What is the time delay (in ms) that the following loop generates? Assume a 25 MHz bus clock. Consider that \texttt{pusha} takes 2 cycles, \texttt{pula} 3 cycles, \texttt{nop} one cycle and \texttt{dbne} 3 cycles.

\begin{verbatim}
ldx #56000
loop:  nop      ; 1 cycle
       nop      ; 1 cycle
       psha     ; 2 cycles
       pula     ; 3 cycles
       psha     ; 2 cycles
       pula     ; 3 cycles
       dbne X, loop ; 3 cycles
\end{verbatim}

\begin{align*}
n \times ntimes &= 15 \\
n \times ntimes \times \frac{1}{25 \times 10^6} &= 15 \times 56000 \times \frac{1}{25 \times 10^6} = 33.6 \times 10^{-3}
\end{align*}

Time Delay = 33.6 ms

(10 pts) After the \texttt{addd} $\$10A0$ instruction, what is the state of D and the following CCR flags: Z, C, V, and N? Does the \texttt{bcs next} instruction branches to 'next'? Yes or no? Why?

\begin{verbatim}
movw #41AC, $\$10A0
ldd #730B
addd $\$10A0
bcs next ...
next: ...
\end{verbatim}

\begin{align*}
V &= c_{16} \oplus c_{15} = 1 \\
C &= c_{16} = 0
\end{align*}

\begin{verbatim}
0x730B = 0 1 1 1 0 1 1 0 0 0 0 0 1 0 1 1 + \\
0x41AC = 0 1 0 0 0 0 0 1 1 0 1 0 1 1 0 0
\end{verbatim}

\begin{verbatim}
0xB4B7 = 1 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1
\end{verbatim}

✓ N flag: MSB of the result. N = 1
✓ C flag: Carry out of the summation. C = 0
✓ Z flag: Test whether the result is 0. Z = 0.
✓ V flag: Overflow when the operation is treated in 2’s complement representation. V = 1

\begin{verbatim}
D \texttt{8B4B7} CCR S X H I N Z V C
\end{verbatim}

✓ \texttt{bcs}: branch if carry set. Since C = 0, then \texttt{bcs next} DOES NOT branch to 'next'.
PROBLEM 6 (20 pts)

- Given the following Assembly code, specify the SP and the Stack Contents at the given times (right after the colored instruction has been executed). SP and the Stack Contents (empty) are specified for the first instruction (LDS #$4000).
- Specify a value in the instruction adda that would make the branch instruction bvs branch to myloop.

```
ROMStart EQU $4000 ; code section
ORG ROMStart

Entry:
_Startup:  LDS  #$4000

myloop:     movw #$FE,2,-SP
            movb #$CA,1,-SP
            leas 3,SP
            adda  #$80
            bvs myloop

forever:    bra forever

; Subroutine
myfun:      pshb
            psha
            leas -2,SP;
            movw #$FAD, SP
            leas 2,SP;
            pula
            pub
            rts
```