(4) **Question 1.** Consider DAC parameters.
Part a) Monotonic
Part b) Resolution
Part c) Precision
Part d) Accuracy

(4) **Question 2.** Write C code that changes the baud rate to 1000 bits/sec.

```
SCI0BD = 500; // n = 8000000/(1000*16)
```

(5) **Question 3.** Use Ohm’s Law, \( V = I \times R \)

\[
1V = R \times 5V \div (10k + R)
\]

\[
10k + R = R \times 5
\]

\[
10k = R \times 4
\]

\[
R = 2.5k
\]

(6) **Question 4.** A measurement system has a range of 0 to 19.9 cm and a resolution of 0.1 cm. Only 1 byte is needed.

Part a) Write assembly code that multiplies the position by 0.5 storing the result back into `position`.

```
ldaa position ; 8-bit unsigned fixed point with 0.1 cm resolution
lsra            ; divide by 2, unsigned
staa position
```

Part b) Write assembly code that adds 2.0 cm to the variable storing the result back into `position`.

```
ldaa position ; 8-bit unsigned fixed point with 0.1 cm resolution
adda #20        ; add 2.0
staa position
```

(4) **Question 5.** Write a C function at receives one character.

```
char SCI0_InChar(void){
    while((SCI0SR1 & 0x20) == 0); // wait for RDRF
    return(SCI0DRL);
}
```

(4) **Question 6.** \( R = (5-2-0.5V)/0.02A = 125 \ \Omega \)

(4) **Question 7.** Draw stack pictures. Assume RegB = $55, RegY=$1234 and RegX = $5678. What is the value in RegX after executing these instructions?

```
pshb
SP-> $55
    stx 2,-sp

SP-> $56
    $78
    $55
    sty 2,sp-

SP-> $??
    $??
    $12
    $34
    $55
    leas 3,sp
    ___$3455___

SP-> $34
    $55
    pulx
```

(6) **Question 8.** Rewrite the assembly subroutine removing the bug.

```
calc TFR D,X
LDY 0,X
LDD #314
EMULS ; need signed
LDX #1000
EDIVS
TFR Y,D
```
(2) **Question 9.** Consider the result of executing the following two 9S12 assembly instructions.

```
ldaa #156
adda #50
```

The carry (C) bit will be 1 because $156 + 206$ does not fit in unsigned 8-bit.

The overflow (V) bit will be 1 because $-100 - 50$ does not fit in signed 8-bit.

(4) **Question 10.** These six events all occur during each output compare 6 interrupt.

D) 1,4,3,2,5,6

(4) **Question 11.** Remember to fetch all object code bytes and push the return address on the stack.

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$4007</td>
<td>$16</td>
</tr>
<tr>
<td>R</td>
<td>$4008</td>
<td>$42</td>
</tr>
<tr>
<td>R</td>
<td>$4009</td>
<td>$00</td>
</tr>
<tr>
<td>W</td>
<td>$3FF3</td>
<td>$0A</td>
</tr>
<tr>
<td>W</td>
<td>$3FF2</td>
<td>$40</td>
</tr>
</tbody>
</table>

(4) **Question 12.** The 10-bit frame = start,1,0,0,0,1,1,0,1,stop. The data is $\$B1$

(24) **Question 13.** In this problem you must use a C data structure that stores this Moore FSM.

**Part a)** Show the C code that defines a linked structure for this FSM.

```c
const struct State{
    unsigned char out;        // 1 means on, 0 means off
    unsigned short threshold; // 0.1 F fixed point
    const struct State *next[2];
};
typedef const struct State StateType;
typedef StateType * StatePtr;
#define ACon  &fsm[0]
#define ACoff &fsm[1]
StateType fsm[2]=
    {0,700,{ACoff,ACon}}, // less than 70 means go to Off
    {1,680,{ACoff,ACon}}  // less than 68 means go to Off
};
```

**Part b)** Write the main that calls `ADC_Init`, initializes the FSM, sets up the OC0, and enables.

```c
StatePtr Pt;
void main(void){
    ADC_Init();
    DDRT |= 0x01;    // PT0 output to AC
    Pt = ACoff;      // initial state
    TIOS |= 0x01;    // activate TC0 as output compare
    TSCR1 = 0x80;    // Enable TCNT, 8MHz
    TSCR2 = 0x07;    // divide by 128, TCNT is 62.5 kHz
    TIE  |= 0x01;    // arm OC0
    TC0  = TCNT+50;  // first interrupt right away
    asm cli            // enable interrupts
    for(;;){};
}
```

**Part c)** Write a C function that samples ADC channel 0 using busy-wait synchronization.

```c
unsigned short ADC_In(void){
    ATD0CTL5 = 0x80;                 // start sequence
    while((ATD0STAT0&0x80)==0){};    // wait for SCF
    return ATD0DR0;
}
```

**Part d)** Write the output compare ISR in C that implements the FSM.

```c
interrupt 8 void TC0han(void){ unsigned short input;
    input = ADC_In();   // Temperature in 0.1F
    if(input < Pt->threshold){
        Pt = Pt->next[0]; // Next state if input less than threshold
```

Jonathan Valvano  December 8, 2010
} else{
    Pt = Pt->next[1]; // Next state if input greater than threshold
}

PTT = Pt->out; // Output depends on the current state
TC0 = TC0+62500U; // every 1s
TFLG1 = 0x01; // acknowledge OC0

(10) Question 14. Reg X stack frame
Part a) Saves Register X, establishes the stack frame, and allocates the locals.
    pshx
    tsx
    leas -6,sp
Part b) Draw a stack picture.
Part c) Show the symbolic binding
left set -6
center set -4
right set -2
Part d) Show code that implements center=100; using Reg X stack frame.
    movw #100,center,x
Part e) Show the assembly code that deallocates the local variables, and restores Reg X.
    leas 6,sp
    pshx
    rts

(15) Question 15. Implement in assembly language a FIFO queue
Part a) Write an assembly subroutine to initialize the FIFO.
Fifo_Init clr Count
    rts
Part b) Write an assembly subroutine that puts one 16-bit element into the FIFO
Fifo_Put tfr d,y
    ldac Count ;0,1,2
    cmpa #2
    beq full
    lsla ;Reg A is 0 or 2
    ldx #Fifo
    sty A,x
    inc Count
    ldd #0 ;success
    bra pdone
full ldd #1 ;full error
pdone rts
Part c) Write an assembly subroutine that gets one 16-bit element from the FIFO.
Fifo_Get tst Count ;0,1,2
    beq empty
    ldd Fifo ;get oldest
    std 0,x ;return by reference
    dec Count
    movw Fifo+2,Fifo ;shift data
    ldd #0 ;success
    bra gdone
empty ldd #1 ;empty error
gdone rts