EE333 Microcontroller Engineering
Oregon Tech Portland, Fall 2012

Homework Assignment #3
Due October 16

Some of the problems have been solved to act as examples. Check your work by assembling and using the simulator.

1. The byte LENGTH contains the (unsigned) length of an edge of a square in centimeters. Calculate the area of the square in units of square inches and store in the the unsigned word AREA.

Using Google Search I found that 1 square centimeter equals 0.15500031 square inches. Since we are limited to integers, I used the Represent program to find out that this is 2500/16129, so if I take the number of square centimeters, multiply by 2500 and then divide by 16129 I'll get the number of square inches.

```
      org   $1000
LENGTH: db   255      ; Test value
AREA: dw   0          ; result goes here
      org   $2000
      ldaa LENGTH     ; get length and square it.
      tfr   a b
      mul
      ldy   #2500     ; multiply by 2500
      emul
      ldx   #16129    ; divide by 16129
      ediv
      sty   AREA      ; store the result
```

The result should be 10078, or 275e in hexadecimal.

2. The textbook has an example of converting Celsius temperature to Fahrenheit. For this problem you need to convert Fahrenheit to Celsius. The word variable FAHR contains the signed temperature. Convert to Celsius and store in the signed word variable CELS.

3. DATA is an array of 5 unsigned bytes, declared as "`DATA: ds 5`". Write into the array so that the first byte contains 0, the second contains 1, the third contains 2, the fourth contains 3, and the fifth contains 4. Use iteration.

I used an indexed addressing mode to access the array.

```
      org   $1000
DATA: ds   5
      org   $2000
      ldx   #DATA     ; address of DATA is in X
      ldaa #5         ; number of bytes to process in A
      clrb            ; value to store in array in B
```

```
loop: stab 1,X+      ; store B at location in X, increment X
      incb           ; increment B
      dbne A loop    ; decrement A and branch if not zero
```

4. DATA is a table of 5 unsigned bytes declared as "`DATA: db 10,21,230,15,7`". Sum the five bytes and store the sum in unsigned word variable SUM. Use iteration.

5. The unsigned word variable DEGREES contains a temperature measurement that we want to display on a two digit readout. The byte variable READOUT needs to have the displayable temperature. Copy DEGREES to READOUT, but if the temperature is 100˚ or more, make the temperature value 99˚.

Here we need to compare DEGREES to 100 and change the temperature if it is greater or equal.

```
      org  $1000
DEGREES: dw    100  ; Need to test at boundary condition
* DEGREES: dw  99   ; The other side of the boundary condition
READOUT: db    0
      org  $2000
      ldd  DEGREES  ; Get the temperature
      cpd  #100     ; compare to 100
      blo  nochange ; If <100, we don't have to change it
      ldd  #99      ; Make the value 99
nochange:
      stab READOUT  ; store the value – a byte but we know that
                    ; the value is in range.
```

The program is run for each of the DEGREES definitions to verify that it is working.

6. An unsigned byte variable LED turns on a warning light when it contains a non-zero value. The warning light is off when it contains a zero value. An unsigned word variable DEGREES contains a temperature measurement. Turn on the warning light if the temperature is 60˚ or below and turn it off if it is 61˚ or above.

7. We want to multiply the unsigned byte values in variables M1 and M2 and store the product in word variable P. Unfortunately our 68HCS12 seems to be broken and the multiply instructions don't work. Write the code to perform this task by adding M2 to P, M1 times. Set P to zero first, of course!

8. The value in unsigned variable POS controls a mechanical positioner and must be in the range 0 through 1023. The desired position has been calculated and stored in signed word variable CTRL. Take the value in CTRL, clamp it so that it is in the range 0 through 1023, and store in POS. We would state the problem in a high level language, POS = min(max(CTRL, 0), 1023).