1. Problem/Objective

The objective of this lab is to write PicoBlaze assembly code to implement a Binary to Binary-Coded Decimal (BCD) conversion. We will be simulating this on the PicoBlaze IDE.

2. Methodology

http://www.idc-online.com/technical_references/pdfs/electronic_engineering/Binary_Coded_Decimal.pdf

BCD is an encoding for decimal numbers in which each digit is represented by its own binary sequence, usually 4 or 8 bits. The main benefit of BCD is that it allows easy conversion to decimal digits for printing or display and faster decimal calculations. The drawbacks are the increased complexity of circuit needed to implement mathematical operations and inefficient encoding since values 10 to 15 are unused. Even with those drawbacks, BCD is still widely used in many financial calculations.

By utilizing BCD, the manipulation of numerical data for display can be greatly simplified by treating each digit as a separate single sub-circuit. This is closer to the physical reality of display hardware. This is shown when trying to interface with a 7 segment display as it is easier if the numeric quantity were stored and manipulated with BCD as pure binary would require a much more complex circuit. Therefore, in cases where the calculations are relatively simple working throughout with BCD can lead to a simpler overall system.

3. Question(s)

Convert 26_{10}, step-by-step, to BCD in the table below. Assume the input is represented using an 8-bit unsigned binary input. See the instructions for an example.

Operation		Special BCD Shift Register		Binary Input
		BCD Digit 1	BCD Digit 0	
Initial				0001 1010
Bit 7			0 (0_{10})	0011 0100
Bit 6			00 (0_{10})	0110 1000
Bit 5			000 (0_{10})	1101 0000
Bit 4			0001 (1_{10})	1010 0000
Bit 3		0 (0_{10})	0011 (3_{10})	0100 0000
Bit 2		00 (0_{10})	0110 (6_{10})	1000 0000
Bit 1		001 (0_{10})	1001 0011 (3_{10})	0000 0000
Bit 0		0010 (2_{10})	0110 (6_{10})	0000 0000

4. Program Code

```
; PBlazeIDE Template
;=========================================================
; data constants
;=========================================================
UPPER_MASK     equ   $0F      ;Mask used to hide 4 MSBs of data in


;=========================================================
; register aliases
;=========================================================
BCD              equ      s0              ;BCD Register 0
data             equ      s1              ;Temp var for manipulations
i                equ      s2              ;Loop Counter
sw               equ      s8              ;Switch Input
sw_in            equ      s9              ;Switch Input Variable
FF               equ      s3              ;Output all 1s if input > 99

;=========================================================
; port aliases
;=========================================================
; input ports
sw_port          dsin     $00

; output ports
led_port         dsout    $80

;=========================================================
; main program
;=========================================================
        call init                         ;run initilization
main_loop:
        comp sw,$64                       ;check if input is > 99
        call nc,over99
        jump output
over99:
        out FF,led_port
        load i,$0
output:
        out BCD,led_port
        comp i,$0                         ;check how many iterations are done
        jump z,done                       ;continuous loop when i = 0
        load data,BCD
        and data,UPPER_MASK               ;mask 4 MSBs of BCD
        comp data,$04                     ;check if BCD > 4
        call nc,add_3                     ;add 3 to BCD
        call find_msb                     ;Find current MSB of sw_in
        comp data,$80                     ;check if MSB is 1 or 0
        call nz,shift_bcd_0
```

```
            comp data,$80
            call z,shift_bcd_1
            sub i,$01
done:
            jump main_loop

;=========================================================
; subroutines
;=========================================================
shift_bcd_0:
             SL0 BCD
             ret

shift_bcd_1:
             SL1 BCD
             ret

add_3:
             ADD BCD,$03
             ret

find_msb:
             load data,sw_in
             and data,$80
             SL0 sw_in
             ret

init:
             load i,$08
             load BCD,$0
             load data,$0
             load FF,$FF
             in sw_in,sw_port
             in sw,sw_port
             ret
```