Microavionics Lab 2 (5067)

Sage Herrin
University of Colorado Boulder

A. Lab Questions

1. Explain the function of the given assembler directives

- **List**
  The function of the List directive is to turn the listing output option on or off. The purpose of the R option is to set the radix value, which can be set to hexadecimal, decimal, or octal. This defaults to hex, but in this lab it is set to decimal.
- **#include**
  The function of the include directive is to read in the specified file as source code. Effectively the same as having entire text of the specified file inserted into the file at the location of the include statement.
  Within the included p18f87k22.inc file, PORTA is defined with an equivalence statement setting it equal to hex value 0F80, which means that it is located in SFR in RAM.
- **CONFIG**
  The CONFIG directive defines a list of configuration bit settings. This includes things such as enabling a watch dog timer or toggling pin multiplexing to on or off.
- **cblock**
  Cblock is used to define a list of named sequential symbols. It is used to assign address offsets to many labels. It is also useful to use for defining constants in program and data memory.
- **org**
  The org directive is used to set the program origin at a specified address. It is often used when code needs to be placed at a particular location.

2.

The **CONFIG FOSC = HS1** option will set the oscillator to medium power mode, which will give the oscillator a frequency range of 4-16 MHz. Switching this configuration to INTI01 would change the Fosc output and I/O pins such that they are set to Fosc/4 output to RA6 and I/O to RA7.

3. At what location in memory is the mainline located at? What about the Initial subroutine?

Utilizing the disassembly listing file, the the mainline program is located at 001C in program memory, and the Initial subroutine is located (or starts at) the location of 0022 in memory.

4.

Checking line 68 of the disassembly listing file, the hex. representation of this instruction is found to be 6E95, which is equivalent to the binary representation 0110 1110 1001 0101. According to the PIC datasheet (page 457), this instruction is used to move data from W to register 'f'. Based on the binary representation for this command stated above, and the Encoding of the command given in the data sheet, the first byte, 0110 1110, shows that the 'a' opcode is set to 0, which tells us that this command specifies that access memory is to be used. The second byte, 1001 0101, specifies the register that W is being moved to.

5.

Repeating the process for the previous problem, now for the CLRF command in the program, the 2 byte hex. representation for this instruction is found to be 6A8C on line 69 of the disassembly listing file. This translates to 0110 1010 1000 1100. According to the PIC datasheet (page 447), this instruction is used to clear the contents of the specified register. If the op code 'a' is set to 0, access bank is selected, and if 'a' is 1 then BSR is selected. Using this info from the data sheet along with the the encoding structure in the data sheet, the first byte, 0110 1010 shows that the 'a' op code is set to 0, thus specifying that the access bank is used. The second byte, 1010 1000 specifies the address that is to be cleared.

6. What is the full RAM address for TRISD and LATD?

According to the data sheet (pages 95-97), and how the access bank is structured in memory, TRISD is located at the address of F95 and LATD is located at F8C.

7.

Noting the values of WREG at the specified point in the program, the watch shows that WREG is set to 0xoo while the program is at address 0022. Once F7 is pressed once, the values of WREG changes from 0X00 to 0xC0. This is because on the previous line, the is a command that loads the 8 bit literal 'k' (or in this case 'B'), into W (MOVLW). The value of 0xC0 appears in the WREG watch because on the previous line with the MOVLW command commands the literal value 'B' of 1100 0000 to be moved to W, and 1100 0000 is equivalent to 0xC0 in hex. Once F7 is pressed again, the value of TRISD changes instead. This is because it has executed the line in the program that contained the command MOVWF TRISD, which is a command to move the value in the working register to the specified location, TRISD in this case.

8.

At this point in the program, once F7 is pressed once, the value of LATD changes. This is due to the command that is executed 'BTG LATD, 0', which is a command that inverts bit 'b' in specified location 'f'. When F7 is pressed several more times, the LATD watch can be seen switching between 0x00 and 0x01 as the bit is inverted each time the command is executed.

9.

At this point in the program, according to the watch values, LATD is equivalent to 0x01, TRISD is equivalent to 0xC0, and WREG is also set to 0xC0. After the program is reset, the only values that changes is that of TRISD, which went from 0xC0 to 0xFF. This is changing because the program is being reset, and the simulator should be simulating what happens in the program when the board is powered on, so TRISD is set to it's POR value, which is 1111 1111 in binary, or 0XFF in hex, according to page 100 in the PIC data sheet.

10.

After the clrf and incf commands were added and the program reset and run in debugging mode, pressing F7 advanced the program by one line and changed the STATUS register value from 0x00 to 0x04. This is because the status affected is listed as Z for this command, hence the change from 0x00 to ox04, which is equivalent to 0100 in Binary. The third bit set to one shows that the Z part of the status was changed from 0 to 1, indicating that the result of an arithmetic operation went from being non-zero to being zero.

11.

After clicking F7 one more time, thus executing the movlw command on line 62, the status register was unchanged. This is because the command movlw has no affect on the status register, so it should read what it read on the previous command, which was 0x04.

12.

After the program was run until the next breakpoint (set at the incf command in the Mainline program), the STATUS register changed from 0x04 to 0x00, implying that an arithmetic operation went from being zero to being non-zero. This is because the incf command affects the Z aspect of STATUS (among other parts of it too), so the Z part of STATUS went from 1 to zero because something went having an arithmetic operation being equal to zero to not being equal to zero, and this is because the incf command is incrementing the value in the working register.

13.

Once F5 was pressed enough times to trigger STATUS to change, STATUS now reads 0x02, and the working register was set to 0x10. The second STATUS bit (DC bit) is now set, indicating that a carry-out from the 4th low-order bit occurred. This makes sense because based on the value of the working register, before this happened the working register was set to 0x0E, equivalent to 1110. After once more cycle of the Mainline loop, the working register was then set to 0x10, equivalent to 0001 0000, thus the DC bit was set.

14.

As the program continues to be cycled until the upper nibble of the working register is changed again. As soon as the upper nibble of the working register changes, the DC bit of STATUS is set for one cycle of the Mainline loop. This is happening because there is going to be a carry-out from the 4th lower-order bit every time the upper nibble of the working register changes because it is longer just changing the first bit or lower nibble of the working register on that execution of the Mainline loop.

15.

After setting the working register to 0x7F and STATUS to 0x00, incrementing through the Mainline loop once more changed STATUS from 0x00 to 0x1A, which is equivalent to 0001 1010. This implies that the Negative bit, Overflow bit, and the Direct Carry bit were all set upon this execution of the loop. Starting with the Negative bit, this being set makes sense since the transition from 0x7F to 0x80 is equivalent to going from -128 to +127, thus the negative bit was changed from 0 to 1. Consequently, because of this change from -128 to +127, the overflow bit is triggered because bit 7 (last bit) was changed from a 0 to a 1, indicating the sign change in the value of the working register due to the overflow from the signed arithmetic. Similar to the last couple of questions, the direct carry bit is set because a carry out of the 4th low order bit occurred due to the change from -128 to +127, thus setting the DC bit of STATUS.

16.

After one more execution of the Mainline loop, STATUS was changed from 0x1A to 0x10. This is becuase the result is still negative (0x10 = 0001 1000 in binary, bit set to 1 is negative STATUS bit), however there was neither overflow nor a direct carry for this operation, so those bits of STATUS were set back to 0.

17.

As F5 is pressed one more time, the working register is incremented again from 0x81 to 0x82 and STATUS remained unchanged, because the values is still negative (0x82 = 0010 0000 = -126), thus the negative STATUS bit is still set.

18.

At this point in the program, before pressing F7, the value of the working register is set to 0x02, and after pressing F7, which executes the DAW command (Decimal Adjust W register), the value of WREG is now set to 0x04. This is becuase the command movwf WREG is moving the value from the working register to the file WREG. Before this command the value of WREG was set to 0x02, then adding WREG to WREG effectively doubled the value in WREG, setting it to 0x04, which is what is happening on this line of the program.

19.

Once the program is run back to the incf line and progress by one line (executin the incf command), the value of WREG is set to 0x05, which increments is by 1 from its previous value 0x04. When F7 is pressed once more, the addwf WREG command is executed, which is adding the working register to itself, or essentially doubling the working register. This is why upon the execution of this instruction the value of WREG goes from 0x05 to 0x0A, which is equivalent to going from 5 to 10. Pressing F7 one more time executes the daw command, short for Decimal Adjust WREG. This essentially is taking whatever value is in the working register and reformatting it into BCD form, or Binary Coded Decimal. This translates to every digit in a number being represented by a binary nibble. For example the number 43 would be represented as 0100 0011 in BCD. This the value of 0x0A (10), is changed to 0x10, which in binary is 0001 0000, which in decimal is 1 and 0, put together to make 10 using BCD format.

20.

After stepping through the program to the lines of interest, the NEGF command changes the working register from 0x04 to 0xFC, and it changes STATUS from 0x00 to 0x10. This is because the NEGF command is negating f, which in this case is the working register, thus the working register goes from 0x04 to 0xFC, equivalent to going from 4 to -4 in decimal. Because of of this, since the output of this arithmetic operation is a negative number, the N bit of STATUS is set.

21.

After rebuilding, running, resetting, and stepping to the line of interest in the program, it's shown that the rlncf command does not change the STATUS value but it does change the WREG from 0xFC to 0xF9. This command is rotating every bit of whatever value is in the working register to the left by 1 without a carry. This changes WREG from 0xFC = -4 = 1111 1100 to 0xF9 = -7 = 1111 1001. Becuse the result is still negative, the negative bit of STATUS remain unchanged.

22.

This time the command rlncf was changed to rlcf and the program was re-run up to the bra line. This time, the value of STATUS went from 0x10 to 0x11, and the WREG changed from 0xFC to 0xF8. This is different than the rlncf command because this command rotates left f through a carry. This explains why now not only the negative STATUS bit is set, but also the carry bit, because the command is rotating f to the left through a carry.

23.

In order to add a variable called 'count', you can simply type 'count' under the cblock line. After this variable is added, a command is added after the rlcf command that would save the working register to the variable count. This command would be movwf count, which would move the value of the working register into the address count, which is specified in the cblock.

24.

Based on the disassembly listing and the operad for the command (movwf count) that was added in the Mainline loop of the program, the variable that is stored in count is stored at the location 0028 in RAM memory.

25.

Using the disassembly listing, the hex representation of the nefg WREG instruction is 6CE8.

26.

Using the disassembly listing, the hex representation of the btg LATD,0 is 708C.

27. Written Lab Questions

COMPUTATIONAL QUESTIONS:

1. What is the two's complement representation of the following decimal numbers? Show your hand calculations to convert these to binary and what method you choose to use. You may use a calculator to check your answers, but you still must show/describe your work!

   (a) −26

   (b) −89

a.) For −26, start by converting to + 26 & finding 2's complement

$\Rightarrow \frac{26}{2} = \frac{\overset{0}{13}}{2} = \frac{\overset{1}{6}}{2} = \frac{\overset{0}{3}}{2} = \overset{1}{1} = \overset{1}{0} = 00011010$

inverting/finding complement $\Rightarrow$ 11100101, adding 1

$\Rightarrow \boxed{1110\ 0110 = -26}$

b) $-89 \Rightarrow +\frac{89}{2} = \frac{\overset{1}{44}}{2} = \frac{\overset{0}{22}}{2} = \frac{\overset{0}{11}}{2} = \frac{\overset{1}{5}}{2} = \frac{\overset{1}{2}}{2} = \frac{\overset{0}{1}}{2} = \overset{1}{0}$

$\Rightarrow$ 01011001, invert $\Rightarrow$ 1010 0110, add 1 $\Rightarrow \boxed{1010\ 0111}$

$\Rightarrow \boxed{-89}$

2. Compute the sum of the following pairs of numbers using 8-bit two's complement arithmetic just like the uC does (convert to binary, do the addition then convert the result back to decimal). Show your hand calculations in binary, you may check your answers with calculator.

   (a) 36, −10

   (b) 16, −77

a.) 36, −10

first using 2's complement on −10 $\Rightarrow \frac{10}{2} = \frac{\overset{0}{5}}{2} = \frac{\overset{1}{2}}{2} = \frac{\overset{0}{1}}{2} = \overset{1}{0}$

$\Rightarrow$ 0000 1010, invert $\Rightarrow$ 1111 0101, add 1 $\Rightarrow$ 1111 0110 = −10

$\frac{36}{2} = \frac{\overset{0}{18}}{2} = \frac{\overset{0}{9}}{2} = \frac{\overset{1}{4}}{2} = \frac{\overset{0}{2}}{2} = \frac{\overset{0}{1}}{2} = \overset{1}{0} \Rightarrow$ 0010 0100 = 36

$\Rightarrow 36 + {}^{-}10 \Rightarrow$

$$\begin{array}{r} \overset{1}{0}\overset{1}{0}\,0\,1\,0\;\;\overset{1}{0}\,1\,0\,0 \\ +\;1\,1\,1\,1\;\;0\,1\,1\,0 \\ \hline 1\,0\,0\,0\,1\;\;1\,0\,1\,0 \end{array}$$

$\Rightarrow 0001\ 1010 = 36 + {}^{-}10 = 26$

b.) 16, -77, $-77 \Rightarrow \frac{77}{2} = \frac{\overset{1}{38}}{2} = \frac{\overset{0}{19}}{2} = \frac{\overset{1}{9}}{2} = \frac{\overset{1}{4}}{2} = \frac{\overset{0}{2}}{2} = \frac{\overset{0}{1}}{2} = \overset{1}{0} \Rightarrow 0100\ 1101,$

invert $\Rightarrow$ 1011 0010, add 1 $\Rightarrow$ 1011 0011 = -11

$\frac{16}{2} = \frac{\overset{0}{8}}{2} = \frac{\overset{0}{4}}{2} = \frac{\overset{0}{2}}{2} = \frac{\overset{0}{1}}{2} = \overset{1}{0} \Rightarrow 0001\ 0000$

$\Rightarrow 16 + {}^{-}77 \Rightarrow$

$$\begin{array}{r} 0\,\overset{1}{0}\,\overset{1}{0}\,1\;\;0\,0\,0\,0 \\ +\;1\,0\,1\,1\;\;0\,0\,1\,1 \\ \hline 1\,1\,0\,0\;\;0\,0\,1\,1 \end{array}$$

$= 1100\ 0011 = 16 + {}^{-}77 = -61$

3. Answer the following questions from Gaonkar: 2.14, 3.4, 3.10, 4.11, 4.13, 4.1SE, 4.2SE, 5.16 (SE questions are to be done by hand and submitted the same as the regular questions. You may choose to use the MPlab simulator to check your answers.)

2.14.) The size of the address bus for the data memory in the PIC18F MCU is 12 bits and can can address 4MB of registers

3.4.) If the BSR register holds byte 01, then the command MOVWF 0X7F, 0 copies the contents of the W register to the register 0X7F in the access bank since the op code is set to 0

3.10.) For the first instruction, MOVLW 0XFA, the W register now is filled with the value 0XFA (1111 1010), and N OV Z & C are set to 1, 0, 0, & 0 respectively

For the next instruction, ADDLW 0X38, the W register now reads 0XFA + 0X38 = 0X32 = 0011 0010 because the W register can only accomodate 8 bits, when the answer should really be 0001 0011 0010 because of this, N OV Z & C will be 0, 1, 0, & 1 respectively due to the overflow & carry necessary in for the operation

4.11) Given the 3 lines of instructions:

Start: MOVLW 0X67

Add LW 0X33

Sleep

a.) W register should be 0X33 + 0X67 = 0XCC

b.) Flags set after addition => N = 1, OV = 1, Z = 0, DC = 0, C = 0

c.) Based on the flags set, the status byte would be 0x18

4.13) calculating sum of 2 bytes

=> 0x92 = 1001 0010, 0xA7 = 1010 0111

=> 0x92 + 0xA7 =>

$$\begin{array}{r} \overset{①}{}1001\,\overset{1}{0}\overset{1}{0}10 \\ +1010\,0111 \\ \hline \boxed{0011\,1001 = 57} \end{array}$$

Based on the instructions/operations set the overflow & carry flags

4.15E)

a.) sum of Byte 1 & Byte 2 => 0X34 + 0X56 =>

$$\begin{array}{r} 34 \\ +56 \\ \hline \boxed{8A} \end{array}$$

b.) Status register is 0x18 (N + OV = 1)

c.) Reg10 contains contents 0X00 since there is overflow

d.) V

4.25E

a.) sum of the bytes =>

$$\begin{array}{r} 35 \\ +2A \\ \hline 5F \end{array}$$

b) Status byte is 0x0000 since there is no carry, overflow, it is not negative, it is positive, & there is no carry-out

c.) Reg10 contains the sum of byte1 + byte2 = 35+2A = 5F because there was no overflow

d.) V

5.16) Instructions to add 2 bytes 78_H & $F2_H$

=> MOVLW 0x78
   ADDLW 0xF2

$$\Rightarrow \begin{array}{r} \overset{1}{} 78 \\ + F2 \\ \hline 6A \end{array}$$

=> carry bit set in status

=> N = 0, OV = 0, Z = 0, DC = , C = 1

for unsigned bytes, the overflow bit will no longer be set, but the carry bit still would be set

QUESTIONS (5067 ONLY)

1. What is the two's complement representation of the following decimal numbers? Show your work in binary.
   (a) −348
   (b) −130

a.) −348 => Start w/ +348

$$\Rightarrow \frac{348}{2} = \frac{\overset{0}{174}}{2} = \frac{\overset{0}{87}}{2} = \frac{\overset{1}{43}}{2} = \frac{\overset{1}{21}}{2} = \frac{\overset{1}{10}}{2} = \frac{\overset{0}{5}}{2} = \frac{\overset{1}{2}}{2} = \frac{\overset{0}{1}}{2} = \overset{1}{0}$$

=> 0000 0001 0101 1100, invert =>

1111 1110 1010 0011, add 1 => 1111 1110 1010 0100 = −348

b.) $-130 \Rightarrow \frac{+130}{2} = \frac{\overset{0}{65}}{2} = \frac{\overset{1}{32}}{2} = \frac{\overset{0}{16}}{2} = \frac{\overset{0}{8}}{2} = \frac{\overset{0}{4}}{2} = \frac{\overset{0}{2}}{2} = \frac{\overset{0}{1}}{2} = \overset{1}{0}$

=> 0000 0000 1000 0010, invert =>

1111 1111 0111 1101, add 1 => $\boxed{\text{1111 1111 0111 1110} = -130}$

2. Compute the sum of the following pairs of numbers using 8-bit two's complement arithmetic (convert to binary, do the addition then convert the result back to decimal). Show your work in binary.

   (a) $-63, -17$

   (b) $-54, 28$

a.) $-63 + -17 \Rightarrow$ Starting w/ 2's complement

$\Rightarrow -63 \Rightarrow \frac{+63}{2} = \overset{1}{\frac{31}{2}} = \overset{1}{\frac{15}{2}} = \overset{1}{\frac{7}{2}} = \overset{1}{\frac{3}{2}} = \overset{1}{\frac{1}{2}} = \overset{1}{0} \Rightarrow 0011\ 1111$, invert $\Rightarrow 1100\ 0000$,

add 1 $\Rightarrow \underline{1100\ 0001 = -63}$

same for $-17 \Rightarrow \frac{+17}{2} = \overset{1}{\frac{8}{2}} = \overset{0}{\frac{4}{2}} = \overset{0}{\frac{2}{2}} = \overset{0}{\frac{1}{2}} = \overset{1}{0} \Rightarrow 0001\ 0001$, invert $\Rightarrow 1110\ 1110$,

add 1 $\Rightarrow \underline{1110\ 1111 = -17}$

$$\Rightarrow -63 + -17 \Rightarrow \begin{array}{r} \textcircled{1}\,1100\ 0001 \\ +\ 1110\ 1111 \\ \hline 1011\ 0000 \end{array} \Rightarrow \boxed{1011\ 0000 = -63 - 17 = -80}$$ w/ carry

b.) $-54, 28$,

same process $\Rightarrow -54 \Rightarrow \frac{+54}{2} = \overset{0}{\frac{27}{2}} = \overset{1}{\frac{13}{2}} = \overset{1}{\frac{6}{2}} = \overset{0}{\frac{3}{2}} = \overset{1}{\frac{1}{2}} = \overset{1}{0} \Rightarrow 0011\ 0110$, invert

$\Rightarrow 1100\ 1001$, add 1 $\Rightarrow \underline{1100\ 1010 = -54}$

$\frac{28}{2} = \overset{0}{\frac{14}{2}} = \overset{0}{\frac{7}{2}} = \overset{1}{\frac{3}{2}} = \overset{1}{\frac{1}{2}} = \overset{1}{0} \Rightarrow \underline{0001\ 1100 = 28}$

$$\Rightarrow -54 + 28 \Rightarrow \begin{array}{r} 1100\ 1010 \\ +\ 0001\ 1100 \\ \hline \end{array}$$
$$\boxed{1110\ 0110 = -54 + 28 = -26}$$

3. Gaonkar 3.19, 4.4SE, 4.5SE, 5.17 (SE questions are to be done by hand and submitted the same as the regular questions. You may choose to use the MPlab simulator to check your answers.)

3.19) The statement can be explained in the way that each 1-word instruction cycle really executes in 2 parts of a cycle, because 1 full cycle is made up of 2 instruction cycles, which consist of fetch & execute

4.4SE) Using same program as 4.1SE, changing byte1 to $F7_H$ & byte 2 to 88_H =>

a.) sum of new data bytes => $\begin{array}{r} ^{1}F7 \\ +\ 88 \\ \hline 17F \end{array}$

b.) Status register contains overflow & carry bits because the sum exceeds the usual 8-bit representation of the working register

=> Status = 0x09

c.) Because an overflow occurred, WREG is set to 0

=> reg 10 = 0

d.) ✓

4.5SE)

After building the given script, the sum of the numbers comes out to be 0x75 in reg11

5.17) For signed numbers, the ov flag & carry flags are both going to be needs, as with signed numbers, negative values are now possible, thus both the overflow & carry bits are necessary / relevant since overflow signals when there is a sign change

Appendix

A variation of this code was used for all questions requiring the use of assembly code, so for the sake of brevity is only included once in the appendix as only very minor changes were made in order to accomplish all problems requiring code.

```
;;;;;;; Lab 2 template for ASEN 4067/5067 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Created: Scott Palo (scott.palo@colorado.edu)
; original: 10-SEP-06
; Updated: Sahe0971@colorado.edu
; Modified: 10-SEP-20
;
; This file provides a basic assembly programming template
;
;;;;;;; Program hierarchy ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Mainline
;   Initial
;
;;;;;;; Assembler directives ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        list  P=PIC18F87K22, F=INHX32, C=160, N=0, ST=OFF, MM=OFF, R=DEC, X=ON
        #include p18f87k22.inc
; After MPLAB X all configuration bits are set in the code
; Use mplab help to understand what these directives mean
CONFIG FOSC = HS1
CONFIG PWRTEN = ON, BOREN = ON, BORV = 1, WDTEN = OFF
CONFIG CCP2MX = PORTC, XINST = OFF


;;;;;;; Variables ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        cblock  0x000       ;

        endc     ; A good place to store variables, none here yet

;;;;;;; Macro definitions ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;; Vectors ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        org  0x0000         ; Reset vector
        nop     ; No operation, wastes one instruction cycle
        goto  Mainline     ; Send program to Mainline code

        org  0x0008         ; High priority interrupt vector
        goto  $             ; $ returns code to the current program counter

        org  0x0018     ; Low priority interrupt vector
        goto  $             ; Returns. Only here to show code structure.



;;;;;;; Mainline program ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
Mainline
        rcall   Initial     ; Initialize everything
Loop
        btg   LATD,0      ; Toggle pin, to support measuring loop time
incf WREG
addwf WREG
negf WREG
rlcf WREG
bra   Loop

;;;;;;; Initial subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; This subroutine performs all initializations of variables and registers.

Initial
reg0 equ 0x00
reg1 equ 0x01
reg2 equ 0x02
reg3 equ 0x03
reg10 equ 0x10
reg11 equ 0x11
movf reg0,0,0
addwf reg2,0,0
movwf reg10,0
movf reg1,0,0
addwfc reg3,0,0
movwf reg11,0
        movlw   B'11000000' ; Move I/O values for PORTD into WREG
movwf   TRISD     ; Set I/O (TRISD)for PORTD
clrf   LATD     ; Drive all OUTPUTS on port D to zero
movlw B'00000001' ; Move literal value of 1 to WREG
        return

        end
```