1 (30) What is the difference between:

1.a Program counter and memory address register? (10)

Program counter stores the address of instruction whereas memory address register stores the address of data. More specifically, PC typically stores the address of the next instruction to be executed.

1.b Accumulator and Instruction Register? (10)

Accumulator store data whereas Instruction Register store instruction. More specifically, Accumulator typically store one of the operand before and the result after arithmetics or logic operations; and the instruction register stores instruction to be executed after it was fetched by the CPU.

1.c General Purpose Register (GPR) based CPU and an Accumulator (ACC) based CPU? (10)

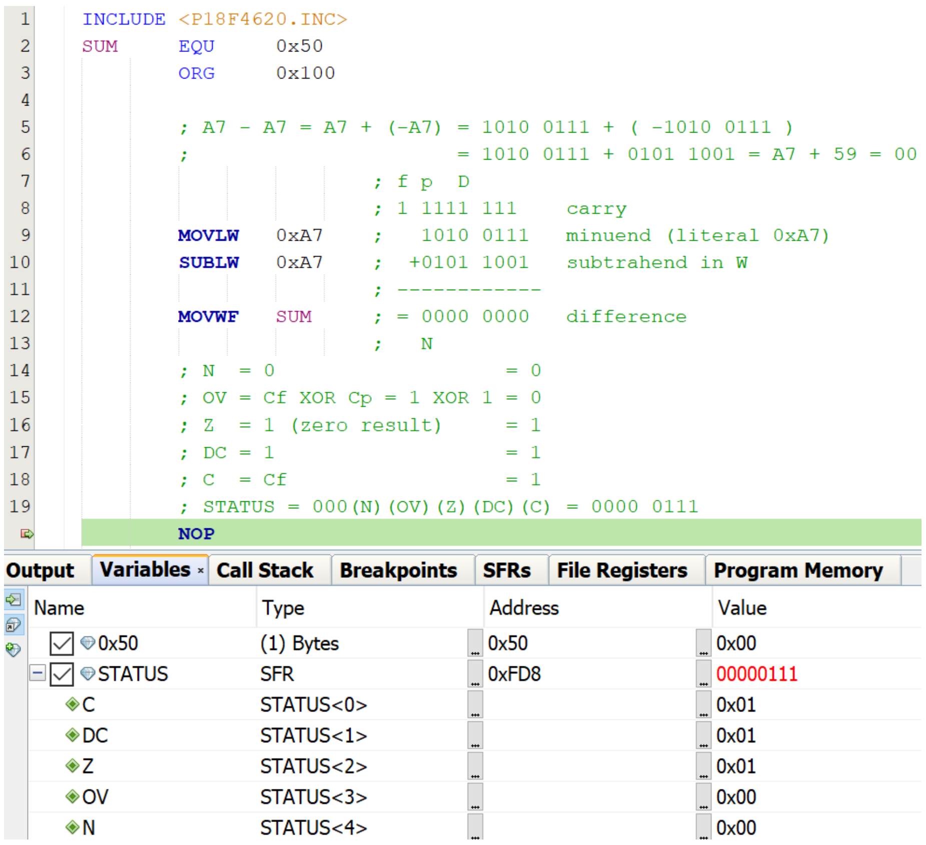
In accumulator-based microcontroller such as the PIC18F, all ALU operations are performed using the Accumulator as one of the data sources, whereas in general-purpose register based microcontroller such as the MSP430, any general-purpose register can be used as an accumulator.

2 (40) Use the PIC18F simulator to calculate the following operations:

Calculate the difference and sum manual calcualtons. Program the problem using assembly code and submit a snap shot of the status register (digit carry, sign, carry, zero, and overflow flags) and the file register showing the result. Did they match?

2.a $A7_{16} - A7_{16}$ (20)

As seen in the following screenshot, the result of manual calculation done in the comments matches the difference stored at 0x50 and the status flags in status register after the calculation is done.



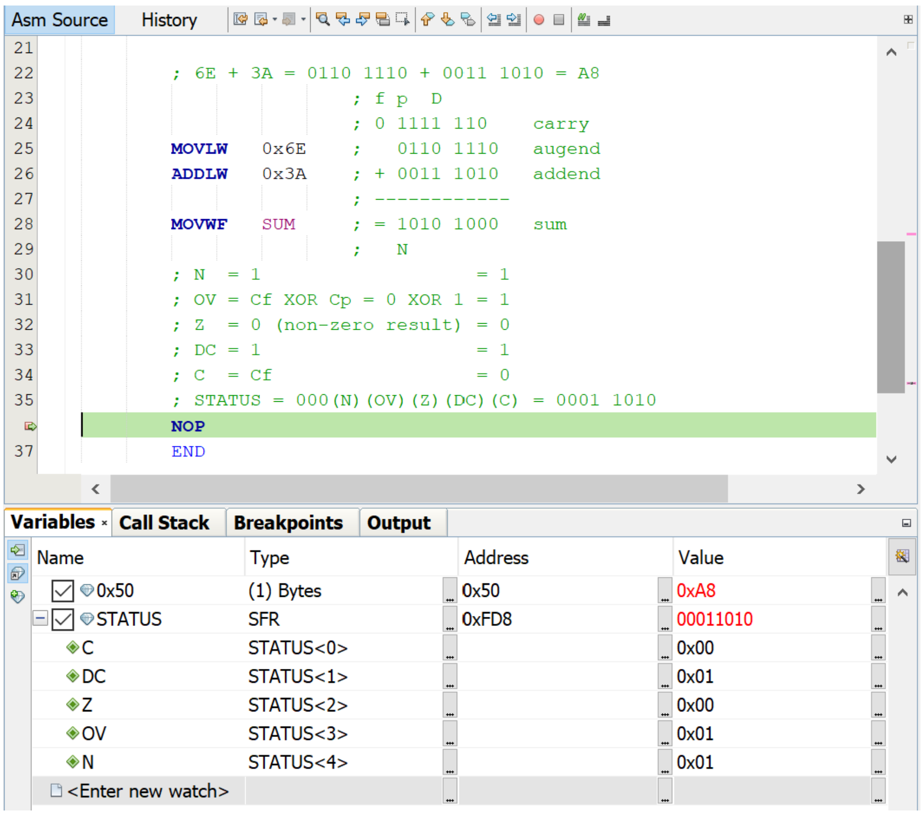
```
INCLUDE <P18F4620.INC>
SUM     EQU     0x50
        ORG     0x100

        ; A7 - A7 = A7 + (-A7) = 1010 0111 + ( -1010 0111 )
        ;                      = 1010 0111 + 0101 1001 = A7 + 59 = 00
                        ; f p  D
                        ; 1 1111 111    carry
        MOVLW   0xA7    ;   1010 0111   minuend (literal 0xA7)
        SUBLW   0xA7    ;  +0101 1001   subtrahend in W
                        ; ------------
        MOVWF   SUM     ; = 0000 0000   difference
                        ;   N
        ; N  = 0                   = 0
        ; OV = Cf XOR Cp = 1 XOR 1 = 0
        ; Z  = 1 (zero result)     = 1
        ; DC = 1                   = 1
        ; C  = Cf                  = 1
        ; STATUS = 000(N)(OV)(Z)(DC)(C) = 0000 0111
        NOP
```

Name	Type	Address	Value
0x50	(1) Bytes	0x50	0x00
STATUS	SFR	0xFD8	00000111
C	STATUS<0>		0x01
DC	STATUS<1>		0x01
Z	STATUS<2>		0x01
OV	STATUS<3>		0x00
N	STATUS<4>		0x00

Figure 1: Manual calculation in comment, state of status register and at 0x50

2.b $6E_{16} + 3A_{16}$ (20)

As seen in the following screenshot, the result of manual calculation done in the comments matches the sum stored at 0x50 and the status flags in status register after the calculation is done.



```
; 6E + 3A = 0110 1110 + 0011 1010 = A8
                ; f p  D
                ; 0 1111 110    carry
MOVLW   0x6E    ;   0110 1110   augend
ADDLW   0x3A    ; + 0011 1010   addend
                ; ------------
MOVWF   SUM     ; = 1010 1000   sum
                ;   N
; N  = 1                   = 1
; OV = Cf XOR Cp = 0 XOR 1 = 1
; Z  = 0 (non-zero result) = 0
; DC = 1                   = 1
; C  = Cf                  = 0
; STATUS = 000(N)(OV)(Z)(DC)(C) = 0001 1010
NOP
END
```

Name	Type	Address	Value
0x50	(1) Bytes	0x50	0xA8
STATUS	SFR	0xFD8	00011010
C	STATUS<0>		0x00
DC	STATUS<1>		0x01
Z	STATUS<2>		0x00
OV	STATUS<3>		0x01
N	STATUS<4>		0x01
<Enter new watch>			

Figure 2: Manual calculation in comment, state of status register and at 0x50

3 (30) What is the difference between PUSH and POP operations in the stack? How is the stack accessed and what effect the access have on addressing the stack memory locations?

In microcontrollers where the stack is accessed from the top (lowest address):

- Stack Pointer (STKPTR) register points to the address of data at the top (lowest address) of the stack.
- PUSH write to the top of the stack whereas POP read from the top of the stack.
- PUSH decrement the STKPTR to point to the next top (lowest address) and write data to that address.
- POP read the item at the top (lowest address) that is being pointed by STKPTR and increment the STKPTR to point to the next data in the stack at the top (lowest address)

In microcontrollers where the stack is accessed from the bottom (highest address):

- Stack Pointer (STKPTR) register points to the address of data at the bottom (highest address) of the stack.
- PUSH write to the bottom of the stack whereas POP read from the bottom of the stack.
- PUSH increment the STKPTR to point to the next bottom (highest address) and write data to that address.
- POP read the item at the bottom (highest address) that is being pointed by STKPTR and decrement the STKPTR to point to the next data in the stack at the bottom (highest address)