3.4 Multi-precision Arithmetic

***Note:** This question shows how multi-precision addition can be done using the ADDC instruction. It also shows how a Stack Pointer based stack frame can be created and employed to store local variables for use within a subroutine. It is important to note that the order of successive addition for multi-word integer depends on whether the numeric value is being stored as Little Endian or Big Endian.*

(1) With reference to Fig. 3.4, for each “?”, give the single VIP mnemonic that will implement the corresponding functionality described by each of the comments shown.

Suggested solutions:

```
Start PSH  [0x100]    ; push the value in N on to the stack                     (m1)
      PSH  #0x101     ; push the address of M1 on to the stack                  (m2)
      CALL SubA       ; make a call to subroutine SubA                          (m3)
         :
         :
SubA  PSH  R2                  ; save value in register R2 on to the stack      (s1)
      SUB  SP,#3               ; create a 3-word stack frame on the stack       (s2)
      MOV  R2,[SP+6]           ; get the contents of N on the stack into R2     (s3)
      MOV  [SP],R2             ; use 1st word (SF1) in stack frame to save value of N (s4)
      MOV  R2,[SP+5]           ; get the address of M1 on the stack into R2     (s5)
      MOV  [SP+1],#0           ; clear 2nd word (SF2) in stack frame            (s6)
      MOV  [SP+2],#0           ; clear 3rd word (SF3) in stack frame            (s7)
Loop  ADD  [SP+1],[R2+0]       ; add lower word of integer and save result in (SF2) (s8)
      ADDC [SP+2],[R2+1]       ; add upper word of integer and save result in (SF3) (s9)
      SUB  [SP],#1             ; decrement value of N in (SF1)                  (s10)
      JEQ  Done                ; jump if N=0            (s11)
      ADD  R2,#2               ; point to next int      (s12)
      JMP  Loop                ; jump back to add again (s13)
Done  MOV  [R2+0],[SP+1]       ; copy result to         (s14)
      MOV  [R2+1],[SP+2]       ; last array item        (s15)
      ADD  SP,#3               ; collapse stack frame   (s16)
      POP   R2                 ; restore orig value of R2 (s17)
      RET
```

	Address	Contents
N	0x100	0x003
M1	0x101	0xFFF
	0x102	0x111
M2	0x103	0x002
	0x104	0x600
M3	0x105	0x222
	0x106	0x100

Memory Map

Fig 3.4 – An incomplete VIP calling program and subroutine

(2) Describe what changes to the program in Fig. 3.4 you would make if the multi-precision integers are stored using the Big Endian format instead. Change only two instructions.

Ans: Since Big Endian has the lower word stored at the higher address, the higher address must be added first using ADD. By storing this result in (SF3) instead of (SF2), there is no need to change instruction at (s14). The higher word in the lower address is then added using ADDC to handle the carry over. Again, storing this result in (SF2) instead of (SF3) means there is no need to change (s15), thus making changes to only two instructions. The following changes must be made:

```
(S8) = Loop  ADD  [SP+2],[R2+1]
(S9) =       ADDC [SP+1],[R2+0]
```

(3) Give the two 12-bit hexadecimal values in memory addresses 0x105 and 0x106 at the end of the execution of the VIP code segment shown in Fig 3.4.

Ans: 0x105=0x223 (lower word) and

0x106=0x812 (upper word)

This is obtained with the following addition:

```
 0x111 FFF
 0x600 002
+0x100 222
 ---------
 0x812 223
 ---------
```