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:

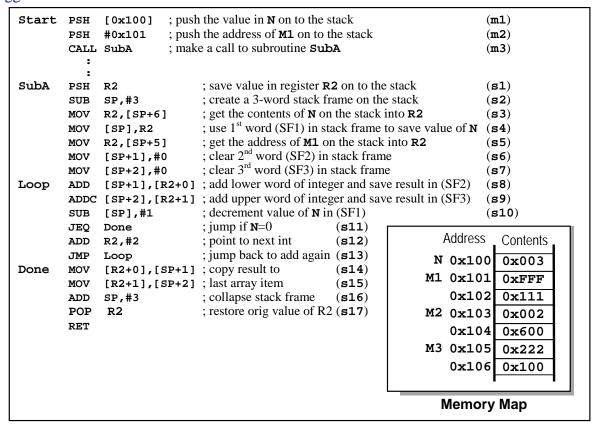


Fig 3.4 – An incomplete VIP calling program and subroutine

(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