Solutions to Optional questions 2.6 in Tutorial #2

2.6 Assembly Programming Exercise – Computing Average

Note to students: *Some important concepts solving this problem will bring across to you are:*

1. *Understand how predetermine looping cycles can be implemented using a countdown counter.*
2. *Understand how unsigned division by 2^n can be implemented using the arithmetic shift right instruction since VIP does not have a logical shift right instruction.*
3. *How consecutive elements in an array can be accessed.*

(1) Write a VIP assembly language program to compute the average value of **eight** numbers in an array of word-sized unsigned integers (i.e. **12-bit unsigned** numbers). The array contains a total to **eight** 12-bit numbers stored in consecutive memory locations starting at address **`0x100`**. Your resulting average (quotient) and remainder should be stored in register **`R0`** and **`R1`**, respectively.

Suggested solution:

```
Start     MOV     R1,#0x100   ; initialize array pointer
          MOVS    R0,#0       ; initialize cumulating reg
          MOVS    AR,#8       ; initialize loop counter for 8 loops
SumLoop   ADD     R0,[R1]     ; fetch data from memory and add to cum reg
          INC     R1          ; increment array pointer to point to next item
          JDAR    SumLoop     ; decrement loop counter and jump if not zero
GetRmndr  MOV     R1,R0       ; make a copy of the sum of all numbers into R1
          AND     R1,#0x007   ; clear all bits except lowest 3 bits to get remainder in R1
GetQuotn  BCSR    1           ; clear Carry flag to make sure 0 enters MSB
          RRC     R0          ; shift right by 1 bit (MSB is now 0)
          RCN     2           ; setup next shift instruction to do 2 bit arithmetic shift
                              ; we can use arithmetic shift as MSB is now 0
          RAR     R0          ; divide sum in R0 by a total of 8, (divide by 2 in the first
                              ; shift and by 4 in the second shift using RAR
                              ; quotient is now in R0 and remainder in R1 - Done
```

Note1:
Using arithmetic shift right (RAR) by 3 bits to divide by 8 is incorrect as we are dealing with unsigned numbers. The most significant bit (MSB) is not a sign bit and should not be replicated during the shift. The VIP instruction set does not have a logical shift right operation, so we need to employ a rotate right with carry to implement the first 1-bit right shift to ensure a '0' moves into the MSB (but C-flag must be cleared to ensure this). Once the MSB of the number is no longer 1, the RAR instruction can be safely used to do the remaining multiple-bit shift right.

Note2:
Notice in the example above, the auxiliary register (AR) was used to implement a more efficient decrement loop as the optimized instruction JDAR does a decrement, test and jump all in one instruction. A more straightforward approach is to use the general register R2 to act as a loop counter and the decrement loop implementation will look something like:

```
Start     :
          MOVS    R2,#8       ; initialize loop counter R2 for 8 loops
SumLoop   ADD     R0,[R1]     ; fetch data from memory and add to cum reg
          INC     R1          ; increment array pointer to point to next item
          DEC     R2          ; decrement loop counter R2
          JNE     SumLoop     ; loop back if R2 not reached zero yet
```

Note3:
There are other ways to implement the divide by 8. An alternative brute force replication method for computing the quotient is:

```
GetQuotn    BCSR    1       ; clear Carry flag to make sure 0 enters MSB
            RRC     R0      ; shift right by 1 bit (divide by 2)
            BCSR    1       ; clear Carry flag to make sure 0 enters MSB
            RRC     R0      ; shift right by 1 bit (divide by 2 again)
            BCSR    1       ; clear Carry flag to make sure 0 enters MSB
            RRC     R0      ; shift right by 1 bit (divide by 2 again ) we have
                            ; now divided by 8 by doing 3 shift right operations
```