Problem EX-4 (3 parts)  

**Part A:** Complete this subroutine which computes the dot product of two vectors of equal length. Assume the first vector starts at base address labeled “A” and the second vector starts at label “B”. Register $1$ contains the vector length in words (# integers), which is greater than 0. Use $2$ to hold the vector index and $3$ to hold the vector length in bytes. Use $5$ and $6$ to hold the current elements of the two vectors, which are multiplied together, and keep the running sum of the products in $4$. Return the dot product in $4$ (i.e., the sum of the products of corresponding elements of the vectors).

<table>
<thead>
<tr>
<th>Label</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DotP:</td>
<td>$# \text{ init vector index to 0}$</td>
<td># init vector index to 0</td>
</tr>
<tr>
<td></td>
<td>$# \text{ init running sum to 0}$</td>
<td># init running sum to 0</td>
</tr>
<tr>
<td></td>
<td>$# \text{ vector length x 4}$</td>
<td># vector length x 4</td>
</tr>
<tr>
<td>Loop:</td>
<td>$# \text{ load current element of A}$</td>
<td># load current element of A</td>
</tr>
<tr>
<td></td>
<td>$# \text{ load current element of B}$</td>
<td># load current element of B</td>
</tr>
<tr>
<td></td>
<td>$# \text{ multiply elements}$</td>
<td># multiply elements</td>
</tr>
</tbody>
</table>
| | $# \text{ put product in }$ | # put product in $5$
| | $# \text{ add product to running sum}$ | # add product to running sum |
| | $# \text{ increment vector index}$ | # increment vector index |
| | $# \text{ loop if index != length}$ | # loop if index != length |
| End: | $# \text{ return to caller}$ | # return to caller |

**Part B:** Create an assembly language function that computes the largest, smallest, and average values in a variable sized array. Here the values are placed in an array in static memory using the `.word` assembler directive. The number of values is implied by the address of the next value in static memory at label `Next:`. An assembly language shell program, shown below, is provided. Place the minimum value in $4$, the maximum in $5$, and the average in $6$.

```
# This program finds the largest, smallest, and average values for
# an integer array.
.data
Array: .word 243, 459, 896, 535, 264, 698, 268, 281, 921, 886
    .word 864, 215, 781, 151, 435, 128, 276, 336, 790, 825
    .word 501, 725, 835, 160, 300, 095, 481, 282, 515, 282
    .word 662, 770, 776, 998, 758, 447, 758, 272, 015, 398
    .word 042, 645, 565, 265, 105, 778, 739, 148, 309, 960
    .word 903, 067, 469, 126, 673, 864, 658, 333, 170, 907
    .word 565, 228, 235, 477, 568, 254, 628, 421, 788, 012
    .word 246, 170, 746, 892, 586, 875, 055, 850, 885, 828
    .word 717, 797, 971, 862, 269, 082, 824, 728, 650, 470
    .word 740, 522, 232, 648, 309, 960
Next: .word 00
.text
F1p2:  addi $1, $0, Array          # set memory base
      # put your program here
      jr $31                     # return to operating system
```
Part C: Create an assembly language function that computes the averages of even and odd numbers in a variable sized array. Here the values are placed in an array in static memory using the .word assembler directive. The number of values is implied by the address of the next value in static memory at label Next:. An assembly language shell program, shown below, is provided. Assume at least one even and one odd value occurs in the list. Your program should place the average of all even numbers in $4 and the average of all odd numbers in $5.

```assembly
* This program the average of all even numbers and the average of all odd
* numbers in an integer array. The even number average is placed in
* $4 and the odd number average is placed in $5.
* Assumes there is at least one even and one odd number in the array.
.data
Array: .word 243, 459, 896, 535, 264, 698, 268, 281, 921, 886
 .word 864, 215, 781, 151, 435, 128, 276, 336, 790, 825
 .word 662, 770, 776, 998, 758, 447, 758, 272, 015, 398
 .word 042, 645, 565, 265, 105, 778, 739, 148, 309, 960
 .word 903, 067, 469, 126, 673, 864, 658, 333, 170, 987
 .word 565, 228, 235, 477, 568, 254, 628, 421, 788, 012
 .word 246, 170, 746, 892, 586, 875, 055, 850, 885, 828
 .word 740, 522, 232, 648, 323
Next: .word 00
.text
EvenOddAvgs:  addi $1, $0, Array       # set memory base
#
put your program here
        jr $31                      # return to operating system
```