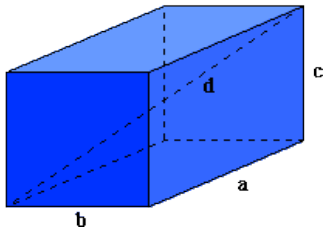
CS 218 – Assignment #1

Purpose: Become familiar with RISC Architecture concepts, the MIPS Architecture, and QtSpim (the MIPS simulator).
Points: 50

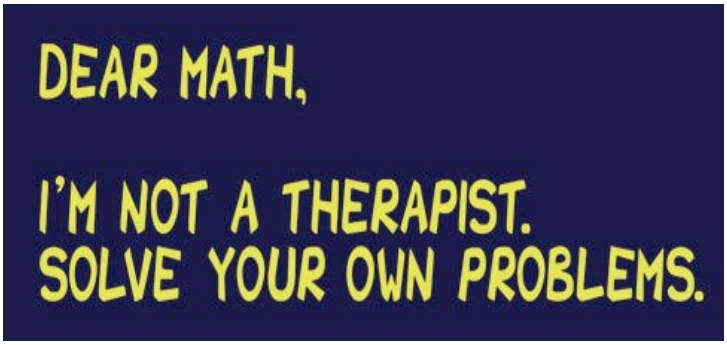
Assignment:



Write a simple assembly language program to calculate the some geometric information for each rectangular parallelepiped in a series of rectangular parallelepipeds. Specifically, the program will find the volume for each of the rectangular parallelepipeds in a set of rectangular parallelepipeds. Once the values are computed, the program should find the minimum, maximum, middle value, and average for the volumes. The formula for the volumes of rectangular parallelepipeds is as follows:

$$volumes[n] = aSides[n] * bSides[n] * cSides[n]$$

Note, for an odd number of items, the middle value is defined as the middle value. For an even number of values, it is the integer average of the two middle values. The data does ***not*** need to be sorted.



The program must display the results to the console window. The volumes should be displayed six (6) per line (they do not need to be justified). Display exactly five (5) spaces before each volumes number. The output should look something like the following (with the correct answers displayed):

```
MIPS Assignment #1
  Program to calculate the volume of each rectangular
  parallelepiped in a series of rectangular parallelepipeds.
  Also finds min, mid, max, sum, and average for volumes.

  Volumes:
     3863654      3079692      2186145      6427036      6940534      1522850
     1811460      2525627      5443181      6973830      -6375336      -3270046

     [...truncated for space...]

  Volumes Min = ?
  Volumes Med = ?
  Volumes Max = ?
  Volumes Sum = ?
  Volumes Ave = ?
```

The printing for the statistical values is includes in the provided template.

Submission:

- All source files must assemble and execute with QtSpim/SPIM MIPS simulator.
- Submit source file
  - Submit a copy of the program source file via the on-line submission
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time (at a maximum rate of 5 submissions per hour).
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All source files must include your name, section number, assignment, NSHE number, and program description. The required format is as follows:

```
#  Name: <your name>
#  NSHE ID: <your id>
#  Section: <section>
#  Assignment: <assignment number>
#  Description: <short description of program goes here>
```

Failure to include your name in this format will result in a reduction of points.

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Assemble	-	Failure to assemble will result in a score of 0.
Program Header	3%	Must include header block in the required format (see above).
General Comments	7%	Must include an appropriate level of program documentation.
Program Functionality (and on-time)	90%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Provided Data:
Use the following data:

```
aSides:             .word       31,    21,    15,    28,    37
                    .word       10,    14,    13,    37,    54
                    .word      -31,   -13,   -20,   -61,   -36
                    .word       14,    53,    44,    19,    42
                    .word      -27,   -41,   -53,   -62,   -10
                    .word       19,    28,    24,    10,    15
                    .word      -15,   -11,   -22,   -33,   -70
                    .word       15,    23,    15,    63,    26
                    .word      -24,   -33,   -10,   -61,   -15
                    .word       14,    34,    13,    71,    81
                    .word      -38,    73,    29,    17,    93

bSides:             .word      101,   132,   111,   121,   142
                    .word      133,   114,   173,   131,   115
                    .word     -164,  -173,  -174,  -123,  -156
                    .word      144,   152,   131,   142,   156
                    .word     -115,  -124,  -136,  -175,  -146
                    .word      113,   123,   153,   167,   135
                    .word     -114,  -129,  -164,  -167,  -134
                    .word      116,   113,   164,   153,   165
                    .word     -126,  -112,  -157,  -167,  -134
                    .word      117,   114,   117,   125,   153
                    .word     -123,   173,   115,   106,   113

cSides:             .word     1234,  1111,  1313,  1897,  1321
                    .word     1145,  1135,  1123,  1123,  1123
                    .word    -1254, -1454, -1152, -1164, -1542
                    .word     1353,  1457,   182,  1142,  1354
                    .word    -1364, -1134, -1154, -1344, -1142
                    .word     1173,  1543,  1151,  1352,  1434
                    .word    -1355, -1037,  -123, -1024, -1453
                    .word     1134,  2134,  1156,  1134,  1142
                    .word    -1267, -1104, -1134, -1246, -1123
                    .word     1134,  1161,  1176,  1157,  1142
                    .word    -1153,  1193,  1184,  1142,  2034

volumes:  .space    220

len:                .word    55

vMin:               .word    0
vMid:               .word    0
vMax:               .word    0
vSum:               .word    0
vAve:               .word    0
```

Note, the **`.space 220`** directive reserves 220 bytes which will be used to store 55 4-byte words.