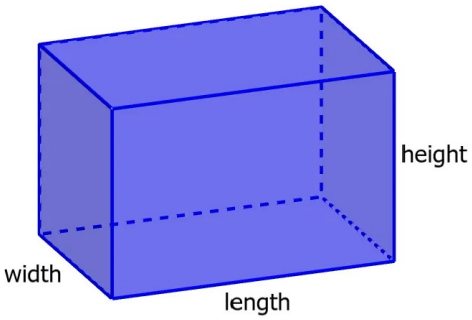
CS 218 – Assignment #5

Purpose: Learn to use arithmetic instructions, control instructions, compare instructions, and conditional jump instructions.
Points: 80

Assignment:

Write a simple assembly language program to calculate some geometric information for each rectangular solid (see diagram to the right). Specifically, the program will find the volume and surface area for each of the rectangular solid in a set of rectangular solids.



Once the values are computed, the program should find the minimum, maximum, estimated median, sum, and average for the volumes and surface areas.

$$volumes[n] = lengths[n] \times heights[n] \times widths[n]$$

$$surfaceAreas[n] = 2 \times lengths[n] \times widths[n] + 2 \times lengths[n] \times heights[n] + 2 \times widths[n] \times heights[n]$$

Since the list is not sorted, we will estimate the median value. Since the list length is odd, the estimated median will be computed by summing the first, last, and middle values and then dividing by 3.

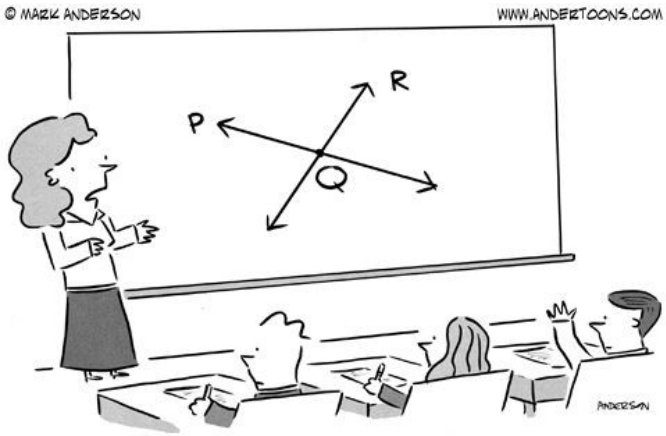
Do ***not*** change the sizes/types of the provided data sets. All data is *signed*. As such, the IDIV/IMUL would be used (not DIV/MUL). Also, CDW/CWD/CDQ must be used (as they are for signed data). The JG/JGE/JL/JLE must be used (as they are for signed data). Of course, these are alternate dimension rectangular solids that allow negative dimensions.

There is no provided main. Create the program source file based on the previous assignments. You may declare additional variables as needed.

Hints:

Pay close attention to the data types. The *lengths[]* array is double-word sized (4 bytes each), the *widths[]* array (2 bytes each) is word sized, the *heights[]* array (1 byte each) is byte sized, the *volumes[]* array is double-word sized and the the *surfaceAreas[]* array is double-word sized.

Consider completing the volumes calculations before attempting the surface area calculations.



"For our purposes there's no crash or explosion. They just intersect."

Submission:

- All source files must assemble and execute on Ubuntu with **`yasm`**.
- Submit source files
  - 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.
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given assignment. 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 loss of up to 10%.

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	5%	Must include header block in the required format (see above).
General Comments	10%	Must include an appropriate level of program documentation.
Program Functionality (and on-time)	85%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Assignment #5 Provided Data Set:

Use the following data declarations for assignment #5. *Note*, the assembler **is** case sensitive.

```
;  Provided Data
lengths       dd      1355,  1037,  1123,  1024,  1453
              dd      1115,  1135,  1123,  1123,  1123
              dd      1254,  1454,  1152,  1164,  1542
              dd     -1353,  1457,  1182, -1142,  1354
              dd      1364,  1134,  1154,  1344,  1142
              dd      1173, -1543, -1151,  1352, -1434
              dd      1134,  2134,  1156,  1134,  1142
              dd      1267,  1104,  1134,  1246,  1123
              dd      1134, -1161,  1176,  1157, -1142
              dd     -1153,  1193,  1184,  1142

widths        dw       367,   316,   542,   240,   677
              dw       635,   426,   820,   146,  -333
              dw       317,  -115,   226,   140,   565
              dw       871,   614,   218,   313,   422
              dw      -119,   215,  -525,  -712,   441
              dw      -622,  -731,  -729,   615,   724
              dw       217,  -224,   580,   147,   324
              dw       425,   816,   262,  -718,   192
              dw      -432,   235,   764,  -615,   310
              dw       765,   954,  -967,   515

heights       db        42,    21,    56,    27,    35
              db       -27,    82,    65,    55,    35
              db       -25,   -19,   -34,   -15,    67
              db        15,    61,    35,    56,    53
              db       -32,    35,    64,    15,   -10
              db        65,    54,   -27,    15,    56
              db        92,   -25,    25,    12,    25
              db       -17,    98,   -77,    75,    34
              db        23,    83,   -73,    50,    15
              db        35,    25,    18,    13

count         dd     49

vMin          dd     0
vEstMed       dd     0
vMax          dd     0
vSum          dd     0
vAve          dd     0

saMin         dd     0
saEstMed      dd     0
saMax         dd     0
saSum         dd     0
saAve         dd     0

; ---------------------------------------------------------------
; Uninitialized data

section       .bss

volumes       resd   49
surfaceAreas  resd   49
```

Note, the ".bss" section is for uninitialized data. The "resd" is for reserve double-words.