CSCI 2500 — Computer Organization
Lab 09 (document version 1.0)
Get your procedures ready!

- This lab is due by the end of your lab session on Wednesday, October 25, 2023.
- This lab is to be completed **individually**. Do not share your code with anyone else.
- You **must** show your code and your solutions to a TA or mentor and answer their questions to receive credit for each checkpoint.
- Labs are available on Mondays before your lab session. Plan to start each lab early and ask questions during office hours, in the Discussion Forum on Submitty, and during your lab session.

1. **Checkpoint 1:** For the first checkpoint, write MIPS code to perform multiplication by addition. Signed integer values to be multiplied should be passed in registers `$a0` and `$a1`. Implement this code as a function labeled `mul`. This function must:
   - Use at least register `$s0` and register `$t0` to store some temporary data. You may also use other registers, as necessary.
   - Make sure that the caller of `mul` does not see the values of `s`-registers change for them. Review slide "Register Usage" from lectures, if necessary.
   - Return the product to the caller. Note that to return a value to the caller, use register `$v0`. The caller must print the value of the product followed by the newline.
2. **Checkpoint 2:** For the second checkpoint, write a MIPS function `det` to find the determinant of a $2x2$ matrix. Recall, that for a $2x2$ matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

   the determinant is defined as $|A| = ad - bc$.

   Your code must:
   - Represent your matrix as a linear array of signed integer elements. E.g., `matrix:␣.word␣1,␣2,␣3,␣0`.
   - Use register `$a0` to specify the base address of the matrix.
   - Return the value of the determinant to the caller. Note that to return a value to the caller, use register `$v0`. The caller must print the value of the determinant followed by the newline.
   - Reuse the `mul` function from the first checkpoint to implement your solution to this checkpoint.
   - Use at least register `$s0` and register `$t0` to store some temporary data in `det`. You may also use other registers, as necessary.
   - Make sure that even if other procedures called by `det` change any of `t`-registers, `det` still works correctly. Review slide "Register Usage" from lectures, if necessary.

3. **Checkpoint 3:** For the third checkpoint, test your `mul` procedure by adding some driver code to the `main` procedure. Your code must:

   - Use the `.word` directive to store the maximum and minimum values of the arguments in memory, e.g.:

```
        .data
arg1_max:     .word 10
arg1_min:     .word -10
arg2_max:     .word 7
arg2_min:     .word -7
```

   - Iterate over all possible combinations of the two arguments, given the limits defined above.
   - Call `mul` with appropriate arguments to find the product of the two arguments.
   - Print the arguments separated by a single space followed by another space, the expected value of the product computed using the `mul` instruction, one more space, and, finally, the actual value of the product returned by `mul`. Here is an excerpt from an example output:

```
-10 -7 70 70
-10 -6 60 60
-10 -5 50 50
-10 -4 40 40
.
.
.
0 6 0 0
0 7 0 0
1 -7 -7 -7
1 -6 -6 -6
1 -5 -5 -5
.
.
.
10 4 40 40
10 5 50 50
10 6 60 60
10 7 70 70
```

   - If the value of the actual product is different from the value of the expected product, the program must print a space and then "Error!" for every test case that is wrong.

```
-10 -7 70 -20 Error!
-10 -6 60 -20 Error!
-10 -5 50 -20 Error!
.
.
.
```