

CS 218 – MIPS Assignment #4

Purpose: Become familiar with the MIPS Instruction Set, and the MIPS standard calling convention, and indexing for multiple dimension arrays.

Points: 80

Assignment

Write a simple assembly language function perform matrix multiplication¹. Each matrix is implemented as a two-dimensional array. The provided main calls two functions:

- Write a MIPS void function, **matrixPrint()**, to display a two-dimensional matrix. The numbers should be printed in a two-dimensional format (see example output). All numbers must be right justified (i.e., lined up on right side). This function will be called by the main and by the **matrixMult()** function.
- Write a MIPS void function, **multMatrix()**, to multiply two matrix's and store the result into a third matrix. Assuming declarations of **MA(iDim, kDim)**, **MB(kDim, jDim)**, and **MC(iDim, jDim)**:

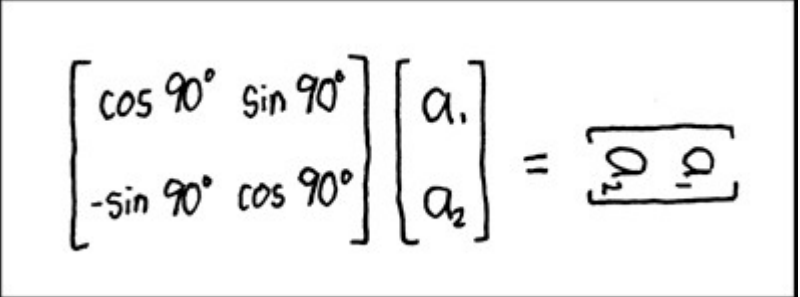
```
for (i=0; i<idim; i++) {
    for j=0; j<jdim; j++) {
        for (k=0; k<kdim; k++) {
            MC(i,j) = MC(i,j) + MA(i,k) * MB(k,j)
        }
    }
}
```

The **multMatrix()** function must use the **matrixPrint()** routine to print the two input matrix's with the appropriate headers. See example output for formatting. The main will print the final resulting matrix.

To compute the address of of an element in a two-dimensional array, use the following formula:

$$\text{board}(\text{row}, \text{col}) = \text{baseAddress} + (\text{rowIndex} * \text{colSize} + \text{colIndex}) * \text{dataSize}$$

You must use this formula. Submissions not using this formula will not be scored.



Source: www.xkcd.com/184

¹ For more information, refer to: https://en.wikipedia.org/wiki/Matrix_multiplication

Submission

When complete, submit:

- A copy of the **source file** via the class web page before class time.

Example Output

The following is an example output for the first two matrix multiplication operations.

```
MIPS Assignment #4
Program to perform matrix multiplication.
```

```
-----
Matrix Set #1
```

```
Matrix MA
```

```
    10    20    30    40
```

```
Matrix MB
```

```
    50
    60
    70
    80
```

```
Matrix MC = (Matrix MA * Matrix MB)
```

```
    7000
```

```
-----
Matrix Set #2
```

```
Matrix MA
```

```
    10    20
    30    30
    50    60
```

```
Matrix MB
```

```
    15    25    35
    45    55    60
```

```
Matrix MC = (Matrix MA * Matrix MB)
```

```
   1050   1350   1550
   1800   2400   2850
   3450   4550   5350
```

Submission

- All source files must assemble and execute with QtSpim/SPIM MIPS simulator.
- 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 3%.

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.
Correct Sort Algorithm	-	Failure to use the provided sort algorithm 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.