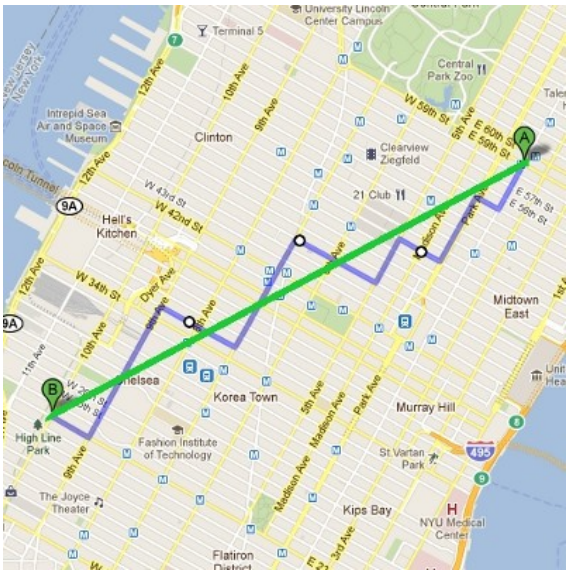
CS 218 – MIPS Assignment #4

Purpose: Become familiar with the MIPS Instruction Set, and the MIPS procedure calling convention, and indexing for multiple dimension arrays.
Points: 90

Assignment:

Write a simple assembly language program to calculate the Manhattan Distance[1] (also referred to as Taxicab Geometry) between two game grids. The Manhattan Distance can be used as a metric to determine how close a current game board configuration is to a final goal or ending configuration.



Each game grid is a two-dimensional array containing numbers representing the current game status. The game grid must be square (rows=columns) and the size is referred to as the *order*. The game board must contain the numbers from **0** to ***order*²-1**. For example, a goal state for a grid of order 3 might be as shown.

1	**2**	**3**
4	**5**	**6**
7	**8**	**0**

The provided main calls the following functions:

- Write a MIPS void function, ***displayBoard()***, that will display a formatted game grid. Refer to the sample output for examples.
- Write a MIPS void function, ***manhattanDistance()***, to determine the Manhattan Distance between the passed goal grid and the current state grid. The distance between two points, (x_0,y_0) and (x_1,y_1), can be computed as follows:

$$\textbf{\textit{distance}} = \sum_{i=0}^{order^2} \left(|x_{i1} - x_{i0}| + |y_{i1} - y_{i0}| \right)$$

  The points represent the grid location. The distance is displayed to the console. Refer to the sample output for examples. For our implementation, the 0 represents an empty space and should be skipped.
- Write a MIPS value returning function, ***validateBoard()***, to validate a game board. The function should return TRUE or FALSE (defined constants). *Note*, A MIPS function returns its result in $v0.

I have a programming joke, but it only works on my computer.

1 For more information, refer to: http://en.wikipedia.org/wiki/Taxicab_geometry

Array Implementation:

In assembly, multi-dimension arrays are implemented as a large single dimension array. The formula for calculating two-dimensional array indexing is:

```
addr[r][c] = baseAddress + (r * colSize + c) * dataSize
```

You must use the formula to access matrix elements. No score will be provided for submissions that do not use this formula. Refer to the text, *MIPS Assembly Language Programming using QtSpim*, Chapter 10, Multi-Dimension Array Implementation for more information.

Example Output:

The following is the example of the output for MIPS assignment #4. *Note*, the output is truncated for space considerations.

```
MIPS Assignment #4
Program to Calculate Manhattan Distance.

***********************************
-----------------------------------
3x3 Goal Board #1

  ---- ---- ----
 |  1 |  2 |  3 |
 |  4 |  5 |  6 |
 |  7 |  8 |  0 |
  ---- ---- ----

-----------------------------------
3x3 Game Board 0

  ---- ---- ----
 |  1 |  2 |  0 |
 |  4 |  5 |  3 |
 |  7 |  8 |  6 |
  ---- ---- ----

Manhattan Distance: 2

-----------------------------------
3x3 Game Board 1

  ---- ---- ----
 |  0 |  1 |  3 |
 |  4 |  2 |  5 |
 |  7 |  8 |  6 |
  ---- ---- ----

Manhattan Distance: 4


               . . .   output truncated  . . .
```

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 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 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.