CSC258 Computer Organization (Winter 2009)
Assignment 3

Due Monday, April 6 at 6:00pm in BA2220

1 [30] Write a program in CSC258 assembly language to implement a judge for the Tic-tac-toe game.

First, the program reads 9 characters from the keyboard, each being either capital "X" or capital "O" or space. The first three characters correspond to the content of the first row on the board, the next three to the central row, and the last three to the bottom row.

After the board state is input, the program then calculates whether there is a line of "X"s (or a line of "O"s) horizontally, vertically, or diagnally. If such a line is found, then output "X wins the game." (or "O wins the game."); otherwise, output "Neither X nor O wins the game."

State your own assumption for invalid inputs (*e.g.* strings including characters other than "X", "O" or space, or a situation containing both a line of "X"s and a line of "O"s, *etc.*), and handle them accordingly in your program.

Here is an example ("_" represents space):

Input:	X_OOXO_XX
Output:	X wins the game.

You must hand in your program with commentary, a sample input/output and your assumptions for invalid inputs on paper in the dropbox in BA2220. You must also submit your program electronically, either by using the submit command on CDF, or by using www.cdf.utoronto.ca/students and click on csc258 and then on submissions. The assignment name is A3Q1 and the file you submit must be named `ttt.ax` The deadline April 6 at 6:00pm is firm; submissions are not accepted after that.

Suggestion: for safety, submit something well in advance, but keep working and resubmit.

2 Write a micro-program (sequence of register transfers) for the CSC258 Computer that implements each of the following instructions. Describe any special details or changes to the basic computer that are needed to add these instructions to the machine. Do not make any changes to the machine that are not needed.

(a) [10] BWI m (Branch With Index) Interpret $[m]$, the content at memory location m, as an index, and jump to the address stored at memory location $m+[m]$.

(b) [10] ACE m (Add and Clear on Error) Add the content at memory location m to the accumulator, and if there is an overflow, set the content at memory location m to 0.