

# CSC258 Computer Organization 2010 winter

## Assignment 3 due Mon.Mar.29 at 5pm in BA2220

1[40] Write a program in CSC258 assembly language to check whether parenthesis pairs ( ) [ ] { } are matched properly. The input is any sequence of characters, using the character # to mean the end of input. The output echoes the input until either an error is detected or the # character is input, and then a message saying either MATCH or ERROR is printed. In the case of error, the offending parenthesis is also printed. For example, if the input is

This (sentence [doesn't]) make {much sense}, but (it has (a lot ( )of parentheses)) in it.#  
then the output is

This (sentence [doesn't]) make {much sense}, but (it has (a lot ( )of parentheses)) in it.  
MATCH

If the input is

This (sentence [doesn't]) make {much sense}, but (it has (a lot ( )of parentheses)) in it.#  
then the output is

This (sentence [doesn't])  
ERROR )

If the input is

This (sentence [doesn't]) make {much sense, but (it has (a lot ( )of parentheses)) in it.#  
then the output is

This (sentence [doesn't]) make {much sense, but (it has (a lot ( )of parentheses)) in it.  
ERROR {

For 80% of the marks you may do an easier version of the problem in which you just match one kind of parenthesis ( ). In this easy version, you just need to count unmatched open parentheses, adding 1 each time you see (, and subtracting 1 each time you see ), always checking that the count does not go below 0, and that it ends at 0. In the hard version with 3 kinds of parentheses, you will need to keep a stack of unmatched open parentheses, adding a new open parenthesis to the top of the stack each time you see one, and popping an open parenthesis from the top of the stack when it is matched.

You must hand in the program and commentary and a sample output on paper in the box 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](http://www.cdf.utoronto.ca/students) and click on csc258 and then on submissions. The assignment name is A3Q1 and the file must be named A3Q1.ax. The deadline is firm; submissions are not accepted after that. (Suggestion: for safety, submit something well in advance, but keep working and resubmit.) The program should be lightly but judiciously commented, and the first line should say your name and student number. The paper commentary on the design should be less than one page.

Suggestion: write the program in the language of your choice, and then translate to assembly language. You can assemble and run your program using ax.c from the course web site and from cdf. You can compile and run ax on any computer, but you should make a final test on cdf because that's where it will be marked.

2 Write a microprogram (sequence of register transfers) for the unoptimized 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)[8] BPA *m* (Branch on Positive Accumulator) If the content of the accumulator, viewed as a two's complement number, is positive (neither negative nor zero), then branch to address *m*, otherwise continue with the next instruction. (The E bit may change.)

(b)[8] MDY *m* (MoDifY - this strange instruction is from a Marconi Myriad computer) The content of memory location *m* is added (two's complement) to the instruction following MDY, and the result is then executed. The instruction following MDY is not changed in memory and is skipped over (not executed next). (The E bit may change.)