CSCI 2500: Computer Organization

Bonus Assignment (5% of Total Grade) C Programming

Date Assigned: April 11, 2015 **Due date:** May 5, 2015

This is a *individual* project (no team programming). The deadline for this assignment is **11 PM, Tuesday, May 5, 2015**. The assignment will not be accepted after that deadline (there is no grace day allowed for this assignment).

Important notes: The C source file for the program must be named `bonus.c` (no header files) Your program will be compiled and tested using the gcc compiler only. This file must be submitted using LMS (email submissions will not be accepted). You must follow the programming and documentation guidelines indicated in the class programming guide posted on LMS. Your program must have several functions in addition to main.

Grading breakdown will be as follows (total of 100 points):

Structure and Documentation: 15 points

Correctness: 85 points

You are required to write a command line C program whose input is a MIPS Assembly Language program and whose output is a listing of the MIPS program with line numbers and an identifier reference table for the MIPS program. Details regarding MIPS programs and the contents of an identifier reference table are given below.

Command Line Details: Suppose the executable version of your program is named bonus.out. The program will be executed by a command line of the following form:

```
bonus.out  inputfile  outputfile
```

In the command line above, the arguments *inputfile* and *outputfile* specify the names of the input file (which contains a MIPS program) and the output file respectively. Your program must produce and write both the MIPS program with line numbers and the identifier reference table to the *outputfile*. Your program must check for the proper number of input arguments. **Programs that don't compile or don't generate the executable will not receive any credit.**

Details Regarding MIPS Programs: Each line of a MIPS program may be a blank line (containing only white-space characters), a comment line (a line that starts with the character '#') or a MIPS statement. Each MIPS statement has an optional label field (terminated by a colon), a mandatory opcode field, an optional operand field consisting of zero or more operands separated by commas, and an optional comment field (starting with the character '#'). The different fields are separated by one or more spaces or tabs.

An identifier in a MIPS program is defined in the source line where the identifier appears as a label and is used in lines where it appears in the operand field. (Note that opcodes are not identifiers). Operands starting with the symbol $ are not identifiers; they represent registers of the MIPS machine.

Example: Consider the following MIPS statement:

```
loop:   la   $s0, intarray   # Load addr. of intarray into register $s0
```

In the above statement, the label field has the identifier `loop`. The opcode field has `la`. The two operands are `$s0` (a register – not an identifier) and `intarr` (an identifier). The operand field is followed by the comment field that starts with the # character. The above statement defines the identifier `loop` and uses the identifier `intarr`.

A identifier reference table for a MIPS program indicates for each identifier, the line number in the source program where the identifier is defined and the line number(s) in the source program where the identifier is used. An example of a MIPS program and its identifier reference table are shown below.

Additional Information on MIPS Programs:

- Every line of a MIPS program is terminated by the newline ('\n') character. There are at most 80 characters (including the newline character) on each line.
- Any line that has '#' as the first character is a comment line.
- In any non-comment non-blank line, the label, opcode, operand and comment fields are separated by one or more spaces or tabs. If such a line has a label, the label will start from the first position (i.e. a line that starts with a space or tab does not have a label).
- The first character of any identifier is a lower or upper case letter or the underscore character; this is followed by zero or more upper/lower case letters, digits or underscore characters. The identifiers are case sensitive (i.e. Loop and loop represent different identifiers). The maximum length of an identifier is 10. You may assume the maximum number of labels that can appear in any MIPS program is 100.
- If the operand field of a MIPS program starts with the single quote character (') or the double quote character ("), then the operand field does not have any identifiers. (The reason is that an operand field starting with the single quote specifies a literal character; an operand field starting with the double quote character specifies a string.)

Errors to be detected: You need to check only for the usual command line errors (e.g. wrong number of parameters on the command line, the input or the output file can't be opened). In such cases, your program must output a suitable error message to standard error (`stderr`) and stop.

Programming Hints:

- Use an array of structs (max size 100) to store the identifier reference table data. Each struct has the following members
  - A char array of size 11 for the identifier name
  - An integer to store the line number where the identifier is defined
  - A linked list where each note of the list stores an integer for the line number where the identifier is used (you may use an array instead of a linked list)
- Don't assume any limit on the number of lines in the input file
- Read the input line by line (use `fgets`)
- For each line of the input file that is not a blank line or comment, use `strtok` to parse the line (i.e. extract the different fields of the instruction)
- You may assume the following about the input files:
  - MIPS program contained in the input file will not contain any errors
  - Every identifier will be defined somewhere in the program but it may be possible that an identifier is not *used* anywhere in the program
  - Identifiers in the program will only be defined once

Example Usage: Suppose the file sample.s contains the following MIPS program:

```
# Sample MIPS Program
           .data      # Data segment begins here
val:       .word      80
_x:        .word      0
           .text      # Text segment begins here
start:     lw    $s0, val
           lw    $s1, _x
           add   $s0, $s0, $s1
           sw    $s0, val        # Store result in val

done:      li $v0, 10
           syscall
```

After we execute your program using the command:

```
bonus.out sample.s sample.lst
```

the contents of the output file sample.lst should be as shown on the next page. Upon examining the output file, you should keep the following in mind:

- The line numbers start at 1
- Blank lines in the input file are echoed to the output file but they are not numbered
- For each identifier, the line numbers where it is used are in increasing order

Example Output File: sample.lst

```
1   # Sample MIPS Program
2               .data       # Data segment begins here
3   val:        .word  80
4   _x:         .word  0
5               .text       # Text segment begins here
6   start:      lw     $s0,  val
7               lw     $s1,  _x
8               add    $s0,  $s0,  $s1
9               sw     $s0,  val        # Store result in val

10  done:       li $v0, 10
11              syscall

Identifier Reference Table

    Identifier  Definition-Line       Use Line(s)
    val         3                    6   9
    _x          4                    7
    start       6
    done        10
```