Lab Assignment 1 Useful Functions

The following functions are provided for your benefit. Feel free to use them if you want. You are not required to use any of this code. It is provided to help you brush up on your C skills and to prevent you from spending a lot of effort writing a parser for your assembler.

Important

To use the code provided below you need to include the following files in your code:

```
#include <stdio.h> /* standard input/output library */
#include <stdlib.h> /* Standard C Library */
#include <string.h> /* String operations library */
#include <ctype.h> /* Library for useful character operations */
#include <limits.h> /* Library for definitions of common variable type characteristics */
```

Parsing Command Line Arguments

```
#include <stdlib.h>

 int
 main(int argc, char* argv[]) {

     char *prgName   = NULL;
     char *iFileName = NULL;
     char *oFileName = NULL;

     prgName   = argv[0];
     iFileName = argv[1];
     oFileName = argv[2];

     printf("program name = '%s'\n", prgName);
     printf("input file name = '%s'\n", iFileName);
     printf("output file name = '%s'\n", oFileName);
}
```

Here's a sample run:

```
tick% assemble ThisGoesIn.asm ThisComesOut.obj
program name = 'assemble'
input file name = 'ThisGoesIn.asm'
output file name = 'ThisComesOut.obj'
```

Question for the reader, what happens when you run this program without any command line arguments? How do you recommend fixing/preventing this problem?

Opening And Closing Files

```
FILE* infile = NULL;
FILE* outfile = NULL;

int
main(int argc, char* argv[]) {

     /* open the source file */
     infile = fopen(argv[1], "r");
     outfile = fopen(argv[2], "w");

     if (!infile) {
       printf("Error: Cannot open file %s\n", argv[1]);
       exit(4);
                }
     if (!outfile) {
       printf("Error: Cannot open file %s\n", argv[2]);
       exit(4);
     }

     /* Do stuff with files */

     fclose(infile);
     fclose(outfile);
}
```

Convert a String To a Number

Hex numbers must be in the form “`x3000`”, and decimal numbers must be in the form “#30”.

```
int
toNum( char * pStr )
{
   char * t_ptr;
   char * orig_pStr;
   int t_length,k;
   int lNum, lNeg = 0;
   long int lNumLong;

   orig_pStr = pStr;
   if( *pStr == '#' )                           /* decimal */
   {
     pStr++;
     if( *pStr == '-' )                         /* dec is negative */
     {
       lNeg = 1;
       pStr++;
     }
     t_ptr = pStr;
     t_length = strlen(t_ptr);
     for(k=0;k < t_length;k++)
     {
       if (!isdigit(*t_ptr))
       {
         printf("Error: invalid decimal operand, %s\n",orig_pStr);
         exit(4);
       }
       t_ptr++;
     }
     lNum = atoi(pStr);
     if (lNeg)
       lNum = -lNum;

     return lNum;
   }
   else if( *pStr == 'x' )       /* hex     */
   {
     pStr++;
     if( *pStr == '-' )                         /* hex is negative */
     {
       lNeg = 1;
       pStr++;
     }
     t_ptr = pStr;
     t_length = strlen(t_ptr);
     for(k=0;k < t_length;k++)
     {
       if (!isxdigit(*t_ptr))
       {
         printf("Error: invalid hex operand, %s\n",orig_pStr);
```

```
        exit(4);
      }
      t_ptr++;
    }
    lNumLong = strtol(pStr, NULL, 16);    /* convert hex string into integer */
    lNum = (lNumLong > INT_MAX)? INT_MAX : lNumLong;
    if( lNeg )
      lNum = -lNum;
    return lNum;
  }
  else
  {
       printf( "Error: invalid operand, %s\n", orig_pStr);
       exit(4);  /* This has been changed from error code 3 to error code 4, see clarificat
  }
}
```

Parsing Assembly Language

Take a line of the input file and parse it into corresponding fields. Note that you need to write the `isOpcode(char*)` function which determines whether a string of characters is a valid opcode.

```
        #define MAX_LINE_LENGTH 255
        enum
        {
           DONE, OK, EMPTY_LINE
        };

        int
        readAndParse( FILE * pInfile, char * pLine, char ** pLabel, char
        ** pOpcode, char ** pArg1, char ** pArg2, char ** pArg3, char ** pArg4
        )
        {
           char * lRet, * lPtr;
           int i;
           if( !fgets( pLine, MAX_LINE_LENGTH, pInfile ) )
                return( DONE );
           for( i = 0; i < strlen( pLine ); i++ )
                pLine[i] = tolower( pLine[i] );

           /* convert entire line to lowercase */
           *pLabel = *pOpcode = *pArg1 = *pArg2 = *pArg3 = *pArg4 = pLine + strlen(pLine);

           /* ignore the comments */
           lPtr = pLine;

           while( *lPtr != ';' && *lPtr != '\0' &&
           *lPtr != '\n' )
                lPtr++;

           *lPtr = '\0';
           if( !(lPtr = strtok( pLine, "\t\n ," ) ) )
                return( EMPTY_LINE );

           if( isOpcode( lPtr ) == -1 && lPtr[0] != '.' ) /* found a label */
           {
                *pLabel = lPtr;
                if( !( lPtr = strtok( NULL, "\t\n ," ) ) ) return( OK );
           }

           *pOpcode = lPtr;

           if( !( lPtr = strtok( NULL, "\t\n ," ) ) ) return( OK );

           *pArg1 = lPtr;
```

```
   if( !( lPtr = strtok( NULL, "\t\n ," ) ) ) return( OK );

   *pArg2 = lPtr;
   if( !( lPtr = strtok( NULL, "\t\n ," ) ) ) return( OK );

   *pArg3 = lPtr;

   if( !( lPtr = strtok( NULL, "\t\n ," ) ) ) return( OK );

   *pArg4 = lPtr;

   return( OK );
}

/* Note: MAX_LINE_LENGTH, OK, EMPTY_LINE, and DONE are defined values */
```

To call `readAndParse`, you would use the following:

```
func()
{
   char lLine[MAX_LINE_LENGTH + 1], *lLabel, *lOpcode, *lArg1,
        *lArg2, *lArg3, *lArg4;

   int lRet;

   FILE * lInfile;

   lInfile = fopen( "data.in", "r" );   /* open the input file */

   do
   {
        lRet = readAndParse( lInfile, lLine, &lLabel,
                &lOpcode, &lArg1, &lArg2, &lArg3, &lArg4 );
        if( lRet != DONE && lRet != EMPTY_LINE )
        {
                ...
        }
   } while( lRet != DONE );
}
```

File Output

To write to your output file, you can use the following:

```
FILE * pOutfile;
pOutfile = fopen( "data.out", "w" );

...

fprintf( pOutfile, "0x%0.4X\n", lInstr );       /* where lInstr is declared as an ir
```