ASSEMBLER PASS 1 REPORT

November 29

2013

Ahmed Nabeel 1450
Gasser Montaser 1473
Mohamed Nader 1671

Introduction

Pass 1 of the Assembler has been implemented by C++. In general, Pass 1 assembles SIC/XE instructions provided by the user in a SRCFILE.txt and outputs a LISFILE.txt to be used in Pass 2 of the assembler.

Instructions for use

1) *SRCFILE.TXT is case insensitive and must be included in assembler's directive.*

2) *Each line in the SRCFILE should represent a single instruction and is allowed to be free-formatted but should ATLEAST contain a single space between each respective string to differentiate between each column (label|opcode|operand).*

3) *Up to 3 strings will be read from any given line, anymore in that same line will be ignored.*

4) *It is not required to provide an END statement at the end of file but you may do so for your convention.*

Input

free-formatted SIC/XE instructions text file named SRCFILE.

Output

a text file named LISFILE that contains assigned addresses per code line and a symbol table.

Data Structures Used

- Structure of type name "Data" that contains the following members:
    - Integer address
      Holds the address of current line.
    - Boolean error
      Is false if current line is free of error.
      Is true if current line contains and error.
    - String label
      Holds the label part of current line.
    - String operand
      Holds the operand part of current line.
    - String opcode
      Holds the opcode part of current line.

- Structure of type name “Values” that contains the following members:
    - Integer format
      Holds the format number of a certain opcode.
    - Integer opcode
      Holds the code (in decimal) of a certain SIC/XE instruction.
- Map of a key value type of string that holds Integers
  Used in constructing the SYMTAB that holds a label and its corresponding address in the memory.
- Map of a key value type of string that holds a structure of type “Values”
  Used in constructing the OPTAB which holds the mnemonic of a SIC/XE instruction and its corresponding format and opcode.

Main Functions Implemented

- LoadMap

  Simply initializes the OPTAB to the correct values.

- ReadInput

  Reads and parses free-formatted input string lines line-by-line from the SRCFILE, tokenizes them and places them into their corresponding variables in structure of "Values" called data_line then finally calls the ProccessData function.

- ProccessData

  Through the help of several helper functions, this function validates (sets error flag) each member variable in data_line, assigns the proper address (in decimal) for each input line through a variable called locctr (location counter). Finally, calls the WriteOutput function.

- WriteOutput

  Simply prints the formatted and parsed data (including errors) onto the console and external text file named LISFILE.

- WriteSYMTAB

  Simply prints the SYMTAB onto the console and external text file named LISFILE.

Helper Functions Implemented

- Validation Functions
    - Formatter

      If called through the ProccessData function, then the current opcode is valid and is redirected to be further parsed by one of the following functions depending on it type:
      check_start;format_2;format_3;resb_resw;word_byte

    - Check_start

      Ensures the proper placement of the START opcode in the text file and deals with any error cases that may arise.

    - Format_2

      Deals with format 2 opcode errors.

    - Format_3

      Deals with format 3 opcode errors.

    - resb_resw

      Deals with the RESB and RESW opcode errors.

    - word_byte

      Deals with WORD and BYTE opcode errors

- Miscellaneous Functions
    - Capitalizer

      Capitalizes string received since the OPTAB uses uppercase for key values of opcode as a convention.

    - To_hex

      Ensures that the string received is hexadecimal and returns it as an integer type in decimal form.

    - To_int

      Ensures that the string received contains numbers only and returns it as an integer in decimal form

Handled Error Cases