**CSc 354 – Assignment #2 – Hamer – Due: 10-9-24**

Write a complete module that will be used as part of your SIC/XE assembler to evaluate the **operand field** of an assembly language statement.

- This must be a separate module from the symbol table module developed in assignment #1.

**Basic Algorithm**

1. read symbols and their attributes one line at a time from a text file named **SYMS.DAT**.
   - SYMBOL VALUE RFLAG
   - exact same process was used with step #1 of assignment #1.
2. read expressions one at a time from the text file whose name was read from the command line.
   - if no file name was specified then prompt the user for the file name.
   - evaluate the current expression.
     - maximum of two values/operands per expression
       - any combination of symbols and numeric literals
     - (+) addition and (-) subtraction are the only supported arithmetic operations
     - use the table given to determine the expressions overall relocatable flag value
     - if the expression begins with = then the operand field contains a literal
       - each unique valid literal is inserted into the literal table
3. display the expression information
   - the required output format is shown – order of attributes is required – see last page
   - display a detailed error message if the expression is invalid
4. display the contents of the literal table
   - the required output format is shown – order of attributes is required – see last page

The expression file will contain one expression per line similar to the following (0 or more leading spaces):

| | |
|---|---|
| GREEN | Simple/Direct Addressing – use RFLAG value |
| @GREEN | Indirect Addressing – use RFLAG value |
| #GREEN | Immediate Addressing – use RFLAG value |
| GREEN,X | Indexed Addressing – use RFLAG value |
| #9 | Immediate Addressing – Absolute value |
| GREEN+YELLOW | VALUE + VALUE and RFLAG + RFLAG |
| GREEN–15 | VALUE – 15 and RFLAG – Absolute value |
| =0cABC | Character Literal – 1 character per byte |
| =0x5A | Hexadecimal Literal – 2 hexadecimal digits per byte |

Rules for evaluating the relocatability of an expression:

- Absolute value – not relative to the starting address of the program – RFLAG is FALSE (0)
- Relative value – relative to the starting address of the program – RFLAG is TRUE (1)

| RFLAG #1 | Operation | RFLAG #2 | Adjusted RFLAG |
|---|---|---|---|
| ABSOLUTE | - | ABSOLUTE | ABSOLUTE |
| ABSOLUTE | - | RELATIVE | **ERROR** |
| ABSOLUTE | + | ABSOLUTE | ABSOLUTE |
| ABSOLUTE | + | RELATIVE | RELATIVE |
| RELATIVE | - | ABSOLUTE | RELATIVE |
| RELATIVE | - | RELATIVE | ABSOLUTE |
| RELATIVE | + | ABSOLUTE | RELATIVE |
| RELATIVE | + | RELATIVE | **ERROR** |

**Literal Table**

- a linked list is used to store each literal along with its associated attributes:
  - literal name – the actual literal expression including = and quotes
    - e.g., =0CABC =0Cabc =0X0F =0X123
    - e.g., =0cABC =0cabc =0x0f =0x123
  - operand value – object code equivalent in hexadecimal
    - e.g., 414243 616263 0F ERROR
  - length in bytes
    - e.g., 3 3 1 ERROR
  - address – initially the literal occurrence within the expression file – eventually the actual address
    - e.g., 0 – first literal encountered, 1 – second literal encountered, …
- the literal table would be a good candidate for an additional standalone module.

Make sure that each module only contains items/operations directly related to that module.

- this applies to all required modules: Symbol Table and Expression Processing
- this applies to all optional modules: Literal Table, String Processing, Error Handling, etc…
  - e.g., the Symbol Table does NOT handle:
    - file processing, expression processing, the literal table, string/character processing, most error handling, etc…

Fully document all parts of your program:

- driver/main program
- modules
  - header (.h) files
  - implementation (.c/.cpp) files
  - C# programs adjust accordingly
- see documentation requirements on the course web site

All output should be in an easy to understand format.

- see the example on the last page of this document.
- do not allow results to scroll off the screen.
  - temporarily pause the screen where appropriate.
- Tera Term Pro uses a default screen size of approximately 20 lines and 80 columns per line.
  - Visual Studio projects adjust accordingly.

All error messages must provide as much detail as possible.

- print out error messages as they are encountered within the expression file
- describe each error in detail as well as display the component or components that generated the error
- make sure not to stop when an error is encountered – process every line in the data file

## Expression Processing Example

**SYMS.DAT**

| | | |
|---|---|---|
| RED: | 13 | TRUE |
| PURPLE: | 6 | FALSE |
| BLACK: | -7 | TRUE |
| PINK: | 9 | TRUE |
| WHITE: | 5 | FALSE |

**Expression File**

```
RED
PURPLE+#17
@BLACK
#WHITE
=0CDEFG
WHITE,X
22
=0X5A
PINK+#3
=0X5A
PINK–#3
@#25+RED
=0C5A
#7
```

When a symbol is encountered its attribute values are determined by looking up the symbol in the symbol table.

**EXPRESSIONS**

| EXPRESSION | VALUE | RELOCATABLE | N-Bit | I-Bit | X-Bit |
|---|---|---|---|---|---|
| RED | 13 | RELATIVE | 1 | 1 | 0 |
| PURPLE+#17 | 23 | ABSOLUTE | 1 | 1 | 0 |
| @BLACK | -7 | RELATIVE | 1 | 0 | 0 |
| #WHITE | 5 | ABSOLUTE | 0 | 1 | 0 |
| WHITE,X | 5 | ABSOLUTE | 1 | 1 | 1 |
| #22 | 22 | ABSOLUTE | 0 | 1 | 0 |
| PINK+#3 | 12 | RELATIVE | 1 | 1 | 0 |
| PINK–#3 | 6 | RELATIVE | 1 | 1 | 0 |
| @#25+RED | 38 | RELATIVE | 1 | 0 | 0 |
| #7 | 7 | ABSOLUTE | 0 | 1 | 0 |

**LITERAL TABLE**

| NAME | VALUE | LENGTH | ADDRESS |
|---|---|---|---|
| =0CDEFG | 44454647 | 4 | 1 |
| =0X5A | 5A | 1 | 2 |
| =0C5A | 3541 | 2 | 3 |

**Notes:**

- @ # ,X apply to the entire expression not an individual operand within the expression
  - E.g., @(OP1+OP2) #(OP1-OP2) (OP1+OP2),X
  - ( ) not part of statement syntax
  - Except would be # associated with a numeric literal: PURPLE+#17
- encountering duplicate literals is not an error
  - only enter valid literal names the first time each unique one is encountered
  - think of them as constants – declared once and used multiple times
- display the contents of the symbol table for debugging purposes