THE PULLET16 ASSEMBLER SYSTEM MANUAL

Functionality Developed By:
Alec Baker, James Cochran, James Thompson, Ethan Weaver, and Chris Walls

CODELINE

FILES:

- codeline.cc
- codeline.h

VARIABLES:

- Boolean:
  - is_all_comment_
    A flag indicating a codeline consists only of comments.
- Integers:
  - linecounter_
    The line counter for this codeline in the source code.
  - pc_
    The program counter of the codeline if one exists.
- Strings:
  - addr_
    The 'addr_' is of a codeline is the last 12 bits of binary bit string.
  - code_
    The machine code corresponding to the codeline.
  - comments_
    Comments in a codeline are indicated by a '*'.
  - error_messages_
    Error messages found in the codeline, set during pass one and two.
  - label_
    The symbols defined on the far-left column of the code line.
  - mnemonic_
    The mnemonic opcode in the codeline.
  - symoperand
    The symbol of which to operated on, i.e. branch to.
- Hex:
  - hex_
    The hex object for ORG, DS, and HEX instructions.

FUNCTIONS:

- Accessors:
  The below functions return the class variable corresponding to their name.
  - GetAddr()
  - GetCode()
  - GetComments()
  - GetErrorMessages()
  - GetHexObject()
  - GetLabel()
  - GetMnemonic()
  - GetPC()
  - GetSymOperand()
- Mutators:
  - SetCommentsOnly()
  - SetErrorMessages()
  - SetMachineCode()
  - SetPC()
- Other Functions:
  - HasLabel()
    Returns a boolean indicating the presence of a label.
  - HasSymOperand()
    Returns a boolean indicating the presence of a symbolic operand.
- IsAllComment()
  - ToString()
    Returns a pretty printed string of the codeline's attributes.

DABNAMESPACE

FILES:

- dabnamespace.cc
- dabnamespace.h

VARIABLES:

- Integers:
  - kMaxMemory = 4096;

FUNCTIONS:

- Other funtions:

        - BitStringToDec()
          Returns the integer value of the passed-in bit string.
        - DecToBitString()
          Returns the string of bits obtained from the passed-in integer value.
        - GetBitsFromMnemonic()
          Converts the bit string code from the textual mnemonic.

HEX

FILES:

- hex.cc
- hex.h

VARIABLES:

- Boolean:
    - is_invalid_
      A flag that tells if a hex operand is valid.
    - is_negative_
      A flag that tells if a hex operand is negative.
    - is_null_
      A flag that tells if a hex operand's 'text_' is empty.
- Integers:
    - value_
      The decimal value of the hex operand.
- Strings:
    - error_messages_
      The error messages set if the hex operand has errors.
    - text_
      The hex value in hex as a string.

FUNCTIONS:

- Accessors:
  The below functions return the class variable corresponding to their name.
    - GetErrorMessages()
    - GetValue()
    - GetText()
- Other Functions:
    - HasAnError()

Returns true if the hex operand is invalid.

  - IsNegative()

    Returns true if the hex operand is negative.
  - IsNotNull()

    Returns true if the hex operand is not null.
  - IsNull()

    Returns true if the hex operand is null.
  - ToString()

    Returns a pretty-printed string of the hex attributes.

PULLET16ASSEMBLER

FILES:

- pullet16assembler.cc
- pullet16assembler.h

VARIABLES:

- Booleans:
  - found_end_statement_

    A flag indicating whether or not the source code contained an end statement.
  - has_an_error_

    A flag indicating whether or not an error was found in the source code.
- Constant Strings:

  The below utility constants are used when generating and reading machine code.
  - kDummyCodeA
  - kDummyCodeB
  - kDummyCodeC
  - kDummyCodeD
- Integers:
  - pc_in_assembler_

    This is a 16-bit program counter, which is initialized to 0 at the beginning of execution, and increments/decrements according to the source code.
  - max_pc_

    The maximum pc value allowed for the 16-bit machine.
- Vectors:

    - codelines_

      The vector of codelines that stores each line of inputed source code.
- Maps:
    - machinecode_

      The map of machine code that stores the machine code generated from the source code.
    - symboltable_

      The map of symbols generated during pass one.
- Sets:
    - mnemonics_

      Container for various valid mnemonic codes in type string.

FUNCTIONS:

- Other Functions:
    - GetInvalidMessage(string, string)

      Catches invalid string symbol and return error message to user.
    - GetInvalidMessage(string, Hex)

      Catches invalid Hex operand and returns error message to user.
    - GetUndefinedMessage()

      Returns an error message stating that passed string symbol is undefined.
    - PassOne()

      Pass one of the assembler takes in the scanner of the source code. With the scanner, it first loops through the source code file, parses the codeline string into definitive codeline attributes according to the length of the line, creates the codeline object, and stores the codeline into the 'codelines_' vector. While this is occuring, the symbol table is set up based on the symbols included in the source code. Several errors are also checked during pass one. Invalid symbols are caught and documented and also checks if a symbol is multiply defined documenting the error. It also checks if the ORG to 'pc_' value is accessible and checks if the DS command has proper allocation.
    - PassTwo()

      Pass two of the assembler does various error checking and generates the final machine code. It does so by looping through the codelines_ vector that was set up in pass one. During each iteration, the mnemonic for the codeline is validated. Any symbol used is checked to make sure it is defined. The address is validated if indirect addressing.

Along with pass one, pass two checks if the ORG to 'pc_' value is accessible and checks if the DS command has proper allocation. If no errors were found during pass two, the machine code is then generated.

- PrintCodeLines()

  Prints out contents of 'codelines_'.
- PrintMachineCode()

  Prints out contents of 'machinecode_'.
- PrintSymbolTable()

  Prints out contents of 'symboltable_'.
- SetNewPC()

  Updates the program counter ('pc_in_assembler_') to new value, if valid.
- UpdateSymbolTable()

  Updates 'symboltable_' with new symbol, sets error flag if passed symbol is multiply defined.

SYMBOL

FILES:

- symbol.cc
- symbol.h

VARIABLES:

- Booleans:
  - is_multiply_

    A flag that states whether or not a symbol was multiply defined
  - is_invalid_

    A flag that states whether or not a symbol was multiply defined
- Integers:
  - location_

    The program counter value when the symbol was discovered.
- Strings:
  - error_messages_

    A string containing the errors messages relative to the symbol.
  - text_

    The actual symbol text itself.

FUNCTIONS:

- Accessors:

  The below functions return the class variable corresponding to their name.
  - GetErrorMessages()
  - GetLocation()
  - HasAnError()

    Checks if either ‘is_multiply’ or ‘is_invalid’ booleans are true.
- Mutators:
  - SetMultiply()

    Sets ‘is_multiply_’ boolean to true.
- Other Functions:
  - CheckInvalid()

    Checks ‘text_’, returns boolean based on whether or not ‘text_’ is valid symbol.
  - ToString()

    Returns pretty-printed string of symbol attributes.