Assembler

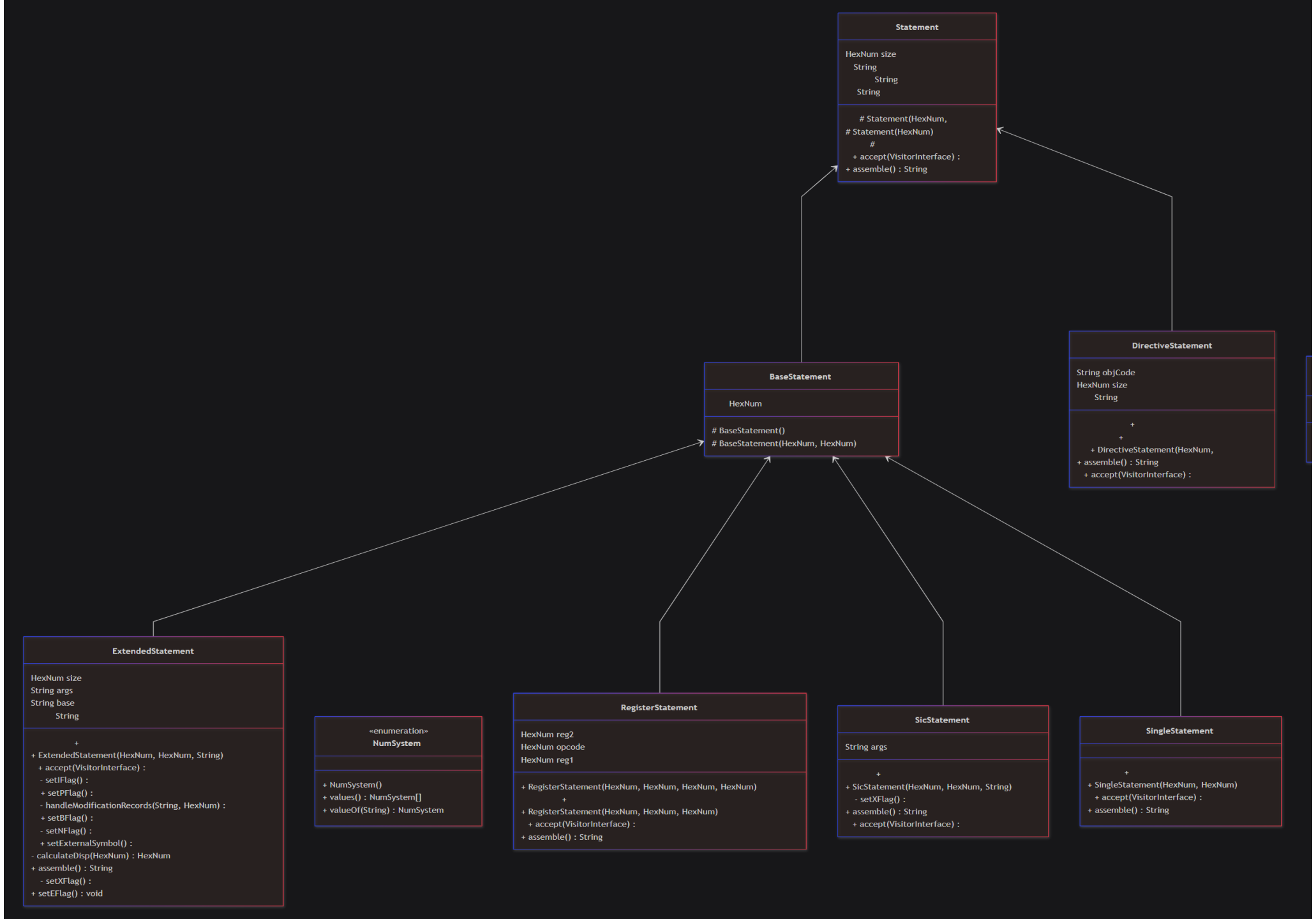
https://salata.software/projects/assembler

Data Storage

- SymTable
  - Stores program block values
  - Stores macro processor definitions
  - Stores all labels organized by their control section
  - Very important to ensure that data exists
- src/main/resources
  - Contains instructions with formats and object code
    - Made it very easy to add new features and keywords
  - Contains register information
  - Contains test files

How lines are represented

- Statement (Abstract)
  - Each Statement class has an assemble method
- DirectiveStatement contains data on assembler directives
- BaseStatement (Abstract)
  - Contains common properties and methods between the following:
- SicStatement assembles lines for an SIC machine
- SingleStatement assembles instructions in format 1
- RegisterStatement assembles instructions in format 2
- ExtendedStatement assembles instructions in format 3/4


Statement
HexNum size
String
String
String
Statement(HexNum,
Statement(HexNum)
#
+ accept(VisitorInterface) :
+ assemble() : String
BaseStatement
HexNum
BaseStatement()
BaseStatement(HexNum, HexNum)
DirectiveStatement
String objCode
HexNum size
String
+
+
+ DirectiveStatement(HexNum,
+ assemble() : String
+ accept(VisitorInterface) :
ExtendedStatement
HexNum size
String args
String base
String
+
+ ExtendedStatement(HexNum, HexNum, String)
+ accept(VisitorInterface) :
- setIFlag() :
+ setPFlag() :
- handleModificationRecords(String, HexNum) :
+ setBFlag() :
- setNFlag() :
+ setExternalSymbol() :
- calculateDisp(HexNum) : HexNum
+ assemble() : String
- setXFlag() :
+ setEFlag() : void
«enumeration»
NumSystem
+ NumSystem()
+ values() : NumSystem[]
+ valueOf(String) : NumSystem
RegisterStatement
HexNum reg2
HexNum opcode
HexNum reg1
+ RegisterStatement(HexNum, HexNum, HexNum, HexNum)
+
+ RegisterStatement(HexNum, HexNum, HexNum)
+ accept(VisitorInterface) :
+ assemble() : String
SicStatement
String args
+
+ SicStatement(HexNum, HexNum, String)
- setXFlag() :
+ assemble() : String
+ accept(VisitorInterface) :
SingleStatement
+
+ SingleStatement(HexNum, HexNum)
+ accept(VisitorInterface) :
+ assemble() : String

Main Stages

- Building
  - Clean up lines
  - Determine which statements to make
  - Manage location counters
  - Manage program blocks
  - Manage expressions
  - Manage literals
  - Manage external definitions and references
  - Returns queues of statements with their exact location
- Writing
  - Executes the assemble method on each statement
  - Writes the header, define, refer, text, modification, and end records
  - Handles all writing logic

Additional Steps

- Builder Builders
  - Handles macro processing
    - Happens externally to builders
  - Handles control sections
    - Each control section is its own builder
    - Hence why we need a builder to build builders
  - Determines which builder to use
    - If our custom SIC flag is set by the programmer, then we use an SIC builder
    - Otherwise, standard builder
- Macro Processors
  - Replaces all parameters with placeholders
  - No processing of statements, just removal of comments
    - Builder handles processing
  - When needed, each parameter placeholder is replaced with given arguments and fed into the builder

Final Notes and Resources

- Full documentation
  - https://salata.software/docs/apidocs/index.html
- UML Diagram
  - https://github.com/sheepman39/Assembler/blob/main/UML.svg
- Source Code with final report in README.md
  - https://github.com/sheepman39/assembler
- Any additional resources used and found are linked in the source code
  - All code, algorithms, and structures were created by me. Resources were used for java documentation, java-specific features, and static analysis.