RISCY - Assembler simulator

Introduction

RISCY is an assembly simulator. It attempts to mimic a load-store ISA (instruction set architecture).

Instruction Set

The instruction set is composed of 32 bits, broken into

- OpCode - First Octet of 4 bits
- Destination Register - Second Octet of 4 bits
- First Register - Third Octet of 4 bits
- Second Register - Fourth Octet of 4 bits
- UNUSED - Fifth Octet of 4 bits
- Offset - Sixth Octet of 8 bits

Opcode

This is essentially the assembly command. This is represented as a map -

```
        {"LOADI", "0001"}, // LOADI R0, #10
        {"ADDRR", "0010"}, // ADDRR R1, R2, R3
        {"ADDRI", "0011"}, // ADDRI R1, #10
        {"BRNCH", "0100"}, // BRNCH <INSTRUCTION>
        {"EQUAL", "0101"}, // EQUAL R4, R5, R6
        {"NQUAL", "0110"}, // NQUAL R4, R5, R6
        {"CLOSE", "1111"}, // CLOSE
```

Destination Register

This is location of the Destination Register. This is represented in a binary format for the location and in code is represented by Registers interface.

First Register

This is location of the First Register. This is represented in a binary format for the location and in code is represented by Registers interface.

Second Register

This is location of the Second Register. This is represented in a binary format for the location and in code is represented by Registers interface.

Offset

This is the memory value of what we are trying to load.

Commands

LOADI

This instruction is used to load a register with a value

```
LOADI R0, #10
# This will load value 10 into register R0
```

ADDRI

This instruction is used to increment existing register with a value

```
ADDRI R0, #25
# This will increment value in R0 register by 25
```

ADDRR

This instruction is used to add values from two registers and save it in a 3rd register

```
ADDRR R1, R2, R3
# This will sum the values of R2 and R3, and load it into R1
```

EQUAL

This instruction is used to compare values between two registers, if equal save 1 in a 3rd register, else 0

```
EQUAL R4, R5, R6
# This will check the values of R5 and R6, if equal will load 1 to R4 else will load 0 to R4
```

NQUAL

This instruction is used to compare values between two registers, if not equal save 1 in a 3rd register, else 0

```
NQUAL R7, R5, R6
# This will check the values of R5 and R6, if not equal will load 1 to R7, else will load 0 to R7
```

Building instructions

Requires only g++ to build. Command:

```
cd <PROJECT_DIRECTORY>
g++ --std=c++17 main.cpp \
    .\architecture\Architecture.cpp \
    .\architecture\Architecture.h \
    .\architecture\Executor.cpp \
    .\architecture\Executor.h \
    .\architecture\Parser.cpp
    .\architecture\Parser.h
    .\architecture\Registers.cpp
    .\architecture\Registers.h
    .\architecture\Stack.cpp
    .\architecture\Stack.h
    -o runRiscy
```

Codebase

This section will document the underlying C++ classes.

RegisterWrapper

TODO

ArchitectureWrapper

TODO

Parser

TODO

Executor

TODO