Group 9 Members:

- Laurenz Luke Gonzales
- Lanbin Wang
- Mahbooba Hashimi

GitHub Link: https://github.com/lgonza32/CSCI6461_Group9_Project.git
Guide/Instructions show in GitHub Readme.md

Goal for Part 0:
Implementing instruction translation through converting assembly instructions as documented in the ISA into 16-bit machine instructions and printed in octal.

This part specifically focuses on:

- Opcode lookup
- Instruction field packing into 16-bits
- Validation and error handling
- Test evidence

Assumptions (as documented in Encoder.java):

- Current basic instruction format
  - [opcode] r, x, address[,I]
  - opcode (6 bits) | r (2 bits) | ix (2 bits) | I (1 bit) | address (5 bits) = 16 bits in total
- Constraints enforced
  - r [0-3]
  - ix [0-3] where 0 means no indexing
  - I [0-1] where it defaults to 0 if omitted
  - Address [0-31]

Project 0 Breakdown:

- src
  - src/part0_assembler
    - opcode_table.java
      - Responsible for mapping instruction mnemonics to numeric opcode values
      - Design:
        - Uses HashMap<String, Integer> to map opcode table provided by ISA document
        - Opcodes stored as integers
      - get(mnemonic)
        - Performs lookup of mnemonic and throws runtime error if opcode does not exist
    - Encoder.java
      - Responsible for converting supported instruction into 16-bit code
      - Design:

        - Operands passed as List<String> to parse
        - RuntimeException handling for debugging
        - Limited to one format for the scope of part 0
    - pack(opcode, r, ix, i, addr)
        - Mask each field to its bit width to prevent overflow
        - Shifts fields into correct position and combine with bitwise OR
    - encodeFormat(mnemonic, args)
        - Validate operand count of 3 or 4 operands
        - Parses numeric operands
        - Checks each field range
        - Calls pack() to generate final instruction
- tests
    - opcode_table_test.java
        - Purpose:
            - Ensure ISA opcode mapping is correct before creating encoding logic
        - Validates that expected opcodes are correct integers and exist
        - Invalidates unknown mnemonics
    - Encoder_test.java
        - Purpose:
            - Ensure that bit packing and ISA field placements are correct