RISC-V Assembler

Description

RISC-V assembler as part of CS2323-Computer Architecture Fall 2024 Lab project. This has limited capabilities, able to handle R,I,S,U,B,J instructions.

Table of Contents (Optional)

If your README is long, add a table of contents to make it easy for users to find what they need.

Implementation

input.s -> lexer -> tokenlist
tokenlist -> parser -> machine code

The input file is tokenized using the lexer which returns a list of tokens.
tokens are stored in structs named `token_struct`

```
typedef struct Token {
    enum {
        TOKEN_INSTRUCTION,
        TOKEN_REGISTER,
        TOKEN_IMMEDIATE,
        TOKEN_LABEL,
        TOKEN_LABEL_REF,
        TOKEN_L_PAREN,
        TOKEN_R_PAREN
    } token_type;
    unsigned int line_number;
    unsigned int character_number;
    unsigned int instruction_number;
    char *value;
} token_s;
```

The list of token is a struct named `token_list_struct`

```
typedef struct TokenList {
    unsigned int index;
    unsigned int size;
    token_s *list;
} token_list_s;
```

During lexing the `size` of `token_struct` is determined.

Next this list of token is set as input to parser, which parses the list instruction by instruction, generating the hex code or showing error message if any.
The parser scans the token_list twice, first to store all the labels and instruction to which it points to, in a hash table, second time to generate code.

Design decisions

- Used a tokenizer instead of a simple string parser to make the assembler more robust.
- Removed comma as token as they were redundant.

Capabilities

- **R format**
  add, sub,xor, or, and, sll, srl, sra, slt, sltu
- **I format**
  addi, xori, ori, andi, slli, srli, srai. slti, sltiu
  lb, lh, lw, ld, lbu, lhu, lwu
  jalr, ecall
- **S format**
  sb, sh, sw, sd

- **B format**
  beq, bne, blt, bge, bltu, bgeu
- **U format**
  lui, auipc
- **J format**
  jal
- **Pseudo instructions**
  nop, mv, not, neg, j
- **Comments**
  (using '#' or ';')
- **Register Aliases**

Issues

- Not able to handle sections.
- Not able to handle all pseudo instructions.
- Error handler only ccapable of checking invalid token error, syntax error and immediate out of bound error.

Challenges faced and solutions

No challenges faced

Tests

Some tests are in test/ which cover the edge cases as well as show the error handling.