RISC-V Simulator Documentation

V. Vishal (cs23btech11064)
M. Sairam Suhas (cs23btech11038)

Introduction

This project is a RISC-V Simulator designed for 64-bit processors, providing functionality similar to Ripes. Users can perform operations such as viewing register values, examining stored memory, tracking executed lines, and setting breakpoints. The simulator also illustrates the stack operations of functions as code executes.

Makefile

The Makefile automates the build process, allowing incremental compilation of the simulator:

- `make all`: Compiles all files into their respective object codes and links them to create the final executable, `riscv_sim`.
- `make <file>.o`: Compiles the specified file to create its object file.
- `make clear`: Removes all object files and the final executable.

Key Functions in `commalgo.cpp`

The following functions are crucial for handling various data types and conversions:

- `int search_label(std::string label, struct store_label label_line[]);`
- `std::string remove_starting_zeros(std::string s);`
- `unsigned int binary_to_decimal(std::string s);`
- `std::string decimal_to_hex(unsigned int decimal);`
- `std::string decimal_to_hex_no(unsigned long long decimal, int no_digits);`
- `std::string signed_decimal_to_hex_no(long long decimal, int no_digits);`
- `int hex_to_deci(std::string s);`
- `std::string hex_to_bi(std::string s);`
- `std::string bi_to_signed_decimal(const std::string& s);`
- `long long int hex_to_deci_64(const std::string& s);`
- `unsigned long long int hex_to_deci_64_unsigned(const std::string& s);`
- `std::string deci_to_bi(int x, int no_of_bits);`
- `std::string register_to_bi(std::string s, int line_counter);`

Core Functions in `risc_sim.cpp`

The main simulation file contains functions that facilitate execution of the RISC-V instructions:

- Commands such as:
  - `load <filename>`
  - `run`
  - `regs`
  - `exit`
  - `mem <addr> <count>`
  - `step`
  - `show-stack`
  - `break <line>`
  - `del break <line>`
- `void check_inst(vector<pair<string, string>> instruction_line, struct regs *x, int *pc , int line_number , stack<stack_label> *show_func , struct store_label *label_line , int data_length);`
- `void run(vector<pair<string, string>> instruction_lines, struct regs *x, int *pc, int *breakpoint , stack <stack_label> *show_func , struct store_label *label_line , int data_length);`
- `void step(vector<pair<string, string>> instruction_lines, struct regs *x, int *pc, int *breakpoint , stack <stack_label> *show_func , struct store_label *label_line , int data_length);`
- `void store_in_mem(string int_value, int *mem_addr, int bytes);`
- `string remove_label(string assembly_instruction, int index, struct store_label *label_line);`
- `vector<pair<string, string>> parse(string filename, struct store_label *label_line, int *data_length);`
- `void print_stack(stack<stack_label> show_func);`
- `void print_mem(int address, int length);`
- `void print_regs(struct regs *x);`
- `void initialize_mem(void);`
- `void initialize_stack(stack<stack_label> *show_func);`
- `void initialize_regs(struct regs *x);`