Interpreter for a subset of MIPS assembly language instructions

Aditya Goel — 2019CS10671
Tushar Singla — 2019CS10410

March 13, 2021

Program Design Specifications:
Input Format

- The program takes in the name of the text file (NOT machine instructions) which consist of MIPS assembly language program, as an arguement to run the interpreter on.
- The allowed instructions in the program are: add, sub, mul, beq, bne, slt, j, lw, sw, and addi.
- The data memory address for lw & sw commands should lie between (524288-1048576).
- The immediate values stored should be integers only.
- For j, bne, beq commands provide the line number to which you want to jump to.
- The total number of instructions run after the completion should not be more than 131,072, for the execution to be successful.

Output Format

- The interpreter prints the Register File contents(32 register values in Hexadecimal format) after executing each instruction.
- After the execution completes, the interpreter also prints statistics like the number of clock cycles and the number of times each instruction was executed. (Assuming that each instruction is executed in one clock cycle).

Strategy & Approach Used

- The code takes a text file as an input and stores every line of that file in a data structure. Then it iterates through that data structure and checks whether a line has some instructions or is a comment.
- The code maintains the count of total number of instructions. Comments are ignored.
- For each instruction the code identifies which instruction is being called and check whether appropriate number of registers, addresses and values have been provided or not. In case of irregular input program, the code throws various errors like:
  - Error: File does not exist or could not be opened
  - Error: Invalid operation
  - Addition Overflow
  - Number of Instructions exceeded the maximum allowed values
  - Line Not Found
  - Invalid Memory address

- After each instruction is executed the complete Register File Contents are printed, i.e. the current values of all the register after each instruction.
- In the end if the execution is successful the code prints total clock cycles and the number of time each instruction was executed.

Test Cases & Corner Cases Handled

1. In case of a comment both, as a separate line or along with those Instructions are completely ignored by the program.
   - A test on input
     #My New Increment Program
     addi $t0, $t0, 1 #Incrementing $t0

     resulted in a successful execution of the program and incremented the value of $t0 by 1 , ignoring any comments that came in between.
   - A test on input
     #My New Increment Program
     #addi $t0, $t0, 1 Incrementing $t0

     resulted in a successful execution of the program with no Instructions to be evaluated.
2. Entering any invalid instruction, other than those specified in the program design specifications prints "Error: Invalid operation" error.
   - A test on input
     blt $t0, $zero, 10

     resulted in an "Error: Invalid operation" error.
3. If results of instructions such as add, sub or addi results in an arithmetic overflow, corresponding errors are raised.
   - A test on input
     addi $t0, $t0, 1
     addi $t0, $t0, 2147483647

     resulted in an "Addition Overflow" error.
4. If at any stage of program, the number of instructions executed is found to exceed 131,072, the execution stops, and print "Number of Instructions exceeded the maximum allowed values" error.
5. If the input to commands such as j, bne, beq is a line which is not present in the Source Code, a "Line Not Found" error is raised.
   - A test on input
     add $t0, $t0, 1
     bne $t0, $zero, 4

     resulted in an "Line Not Found" error.
6. Entering a memory address beyond the allowed bounds of (524288-1048576) or a memory address that is not a multiple of 4, prints "Invalid Memory address" error. While If a memory address being accessed has never been initialised to some value, prints an "No Value Found At The Given Address" error.
   - A test on input
     sw $t0, 1048580

     resulted in an "Invalid Memory address" error.
   - A test on input
     lw $t0, 524288

     resulted in an "No Value Found At The Given Address" error.
7. Entering an input with register names other than the 32 registers specified, prints an "Error: Invalid register" error. Also entering non-32 bit integer values in instructions such as addi, prints an "Error: Given value is not a valid number" or "Error: Out of Range Integer" error respectively.

– A test on input
  addi $m0, $m0, 1

  resulted in an "Error: Invalid register" error.

– A test on input
  addi $t0, $t0, abc

  resulted in an "Error: Given value is not a valid number" error.

– A test on input
  addi $t0, $t0, 2147483648

  resulted in an "Error: Out of Range Integer" error.