MIPS Interpreter

By- Pranay Gupta (2019CS10383)
Divyanka Chaudhari (2019CS40429)

Working of the simulator

The simulator has registers r0-r9, s0-s9, t0-t9, v0-v1 which can be used for various operations. On running the code on an input text file present in the same directory, the code prints the register contents after each instruction along with the instruction number. At the end of the code, the total number of instructions and the number of times each operation was called is written. Of the 2^20 bytes, 100000 bytes of memory have been assigned to instruction memory, since each instruction takes 4 bytes of memory, the total no. of instructions that could be computed are 25000 else an error is thrown. The rest of the memory is allotted to data and trying to access a value outside data memory throws an error. The j label is given a value that represents the line number the code must jump to. Similarly for the bne and beq branches. Finally, the lw and sw commands store data in memory address starting from 0 and going upto 237144 since each word requires a memory of 4 bytes. Finally, trying to access a memory address that has not initialized throws a "Memory address not initialized error". You can write comments in the code on a new line that begin with a '#'. On entering a branch change more than the number of lines in the code, the code ends there giving the output. Only the first syntax error is printed and not all of them. On entering an empty input file or wrong file name, the initial commands initialized to 0 are printed.

Approach

We have first defined a function that prints the hex values of a set of 32 registers. This function is called after every instruction. Moreover, we have an str to int function that also checks if the given input contains only numerals as input else it throws an error. We initialize the 32 values of the registers to 0 in a vector. We have a hashmap for the data memory and a vector for the instruction memory that takes care of the appropriate memory considerations for both of them. In the main code, we read data from a file and store all the instructions in an instruction set. We read the code one by one in lines and check what the user has asked to be computed and then do the appropriate calculations over there. Moreover, we also throw syntax errors if the user tries to enter a syntactically incorrect input. The registers are printed after each iteration.

Test Cases

1. Syntax errors-

```
add2, 31
addi $r2, $r2, -32
mul $r3, $r2, $r1
j 15

sw $r3, 83910380
lw $r4, 2
```

```
mul $r3, $r2, $r1
addi $r2, $r2, -32
```

Output: Invalid Syntax at line 1

```
addi $r1, $r2, 31
addi $r2, $r2, -32
mul $i3, $r2, $r1
j 15

sw $r3, 83910380
lw $r4, 2

mul $r3, $r2, $r1
addi $r2, $r2, -32
```

Since i3 is not a valid register name, the code gives a Syntax error at line 3 after printing the register values for the fisrt 2 instructions.

```
addi $r1, $r2 31
addi $r2, $r2, -32
mul $i3, $r2, $r1
j 15

sw $r3, 83910380
lw $r4, 2

mul $r3, $r2, $r1
addi $r2, $r2, -32
```

Since a comma is missing in the first line, the code throws a syntax error at line 1

```
addi $r1, $r2 31sdsds
addi $r2, $r2, -32
j 31
```

Output: Invalid Syntax at line 1

```
a
addi $r2, $r2, -32
```

Output: Invalid Syntax at line 1

2. Data memory overflow and accessing uninitialized memory

```
addi $r1, $r2, -3221
addi $r2, $r2, -32
#Addition of 2 registers
```

```
add $r3, $r2, $r1
lw $r1, 0
```

Output: Trying to access uninitialised memory at line:5

```
addi $r1, $r2, -3221
addi $r2, $r2, -32
#Addition of 2 registers
add $r3, $r2, $r1
sw $r1, 322443
```

Output: Data Memory overflow at line:5

3. Instruction memory overflow
   The input file t1.txt containing 50000 lines of instructions was run
   Output: Instruction Memory Overflow

4. Working of the code

```
addi $r1, $r2, -3221
addi $r2, $r2, -32
#Addition of 2 registers
add $r3, $r2, $r1
sw $r1, 322
lw $r4, 322
mul $r5, $r4, $r3
```

Final output is saved in a file called out1.txt

```
addi $r1, $r2, -3221
j 6
addi $r2, $r2, -32
#Addition of 2 registers
add $r3, $r2, $r1
sw $r1, 322
lw $r4, 322
mul $r5, $r4, $r3
```

Final output is saved in a file out2.txt

```
addi $r1, $r2, -3221
bne $r1, $r2, 6
addi $r2, $r2, -32
#Addition of 2 registers
add $r3, $r2, $r1
sw $r1, 322
```

```
lw $r4, 322
mul $r5, $r4, $r3
```

Final output is saved in a file out3.txt

```
addi $r1, $r2, -3221
bne $r1, $r2, 6
addi $r2, $r2, -32
#Addition of 2 registers
add $r3, $r2, $r1
sw $r1, 322
lw $r4, 322
addi $r5, $r6, 1
slt $r2, $r1, $r5
mul $r5, $r4, $r3
```

Final output is saved in a file out4.txt