COL216: ASSIGNMENT – 1

Student 1: Bommakanti Venkata Naga Sai Aditya **Entry No. 1:** 2019CS50471

Student 2: Alladi Ajay **Entry No. 2:** 2019CS10323

APPROACH

Input:

Input to our program is a text file with MIPS instructions. Input file is given through command line. Execute using the following instruction:

`./2019CS50471_2019CS10323_Ass3 “path to inputfile”`

Algorithm:

Step – 1: Reading the inputs and storing it in a vector:

Our first go through each line of the input file and parse each instruction. It identifies the command in the instruction, registers in the instruction, values in the instruction and stores these in a special data structure. And this data structure is stored in vector PC.

Step – 2: Executing the Instruction:

Next, our program goes through each instruction in PC and executes the instruction. If the instructions has any synatx errors our programs flags them. Syntax of the instructions is as follows:

add reg1, reg2, reg3

sub reg1, reg2, reg3

mul reg1, reg2, reg3

beq reg1, reg2, line no

bne reg1, reg2, line no

slt reg1, reg2, reg3

j line no

lw reg1, offset(reg2)

sw reg1, offset(reg2)

addi reg1, reg2, reg3

Here: reg1, reg2, reg3 are the registers. In branch and jump instrcutions, line number to which we should jump should be provided.

Memory:

Our program takes a single memory block with 2^20 bytes. When we read the instructions, our memory block gets updated as Instructions memory and data memory. After all instructions are loaded, the remaining space is used for data memory. We considered the whole memory block as chunks of 4 bytes. So, data should be written only at memory addresses which are multiples of 4, because each integer requires 4 bytes. So, in instructions lw and sw, offset + value(reg2) should be a multiple of 4. If this convention is not followed, our program flags an error. All the data memory is initialized to zero.

Output:

Register values after each instruction are printed on the terminal. And the no of clock cycles and no of times each instructions is executed are printed on terminal.

TESTING

General Case: All the instructions follow the given syntax as mentioned above.

We have found the following possible corner cases:

1) Syntax errors: If any instruction didn't follow the given syntax, then our program will flag an error along with the line number in which the error occured.
2) Invalid Memory Address: If the memory address provided by the user exceeds the given limit or lies within the memory block of instructions, then our program will flag an error saying that invalid memory address is entered. Also, lw and sw operations will be performed only if the given address is a multiple of 4. Because we assumed the memory block as chunks of 4 bytes.
3) Invalid line number in branch statements: If the line number entered by the user in branch instructions is not valid, then our program flags an error.
4) Invalid command or register: If any other command or register other than the mentioned ones is given, then our program will flag an error.
5) Modifying zero register: Our program flags an error if the user tries to modify zero register. (Zero register can't be modified)

Test Cases:

Testing Strategy: We computed the output manually and verified it with the output

General Case :

1) addi, add, sub, mul

Input:

```
1: addi $t0, $zero, 5
2: add $t1, $t0, $t0
3: sub $t2, $t0, $t1
4: mul $t3, $t2, $t1
```

Output:

```
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 5, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 : 0,
s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp :
0, sp : 0, fp : 0, ra : 0]


[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 5, t1
: 10, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 : 0,
s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp :
0, sp : 0, fp : 0, ra : 0]


[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 5, t1
: 10, t2 : -5, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]


[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 5, t1
: 10, t2 : -5, t3 : -50, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]


Clock Cycles: 4

Number of times instructions in line 1, executed is 1

Number of times instructions in line 2, executed is 1

Number of times instructions in line 3, executed is 1

Number of times instructions in line 4, executed is 1
```

2) addi, slt, j, add, sub

Input:

```
1: addi $t0, $zero, 20
2: addi $t1, $zero, 10
3: slt $t3, $t0, $t1
4:
5: j 7
6: add $t5, $t0, $t1
7: sub $t4, $t3, $t1
```

Output:

```
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 20,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 20,
t1 : 10, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 20,
t1 : 10, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 20,
t1 : 10, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 20,
t1 : 10, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 20,
t1 : 10, t2 : 0, t3 : 0, t4 : -10, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2
: 0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0,
gp : 0, sp : 0, fp : 0, ra : 0]

Clock Cycles: 5
Number of times instructions in line 1, executed is 1
Number of times instructions in line 2, executed is 1
Number of times instructions in line 3, executed is 1
Number of times instructions in line 5, executed is 1
Number of times instructions in line 6, executed is 0
Number of times instructions in line 7, executed is 1
```

3) addi, add, bne, beq, sub

Input:

```
1: addi $s0, $zero, 15
2: addi $s1, $zero, 10
3: addi $s2, $zero, 1
4: add $s1, $s1, $s2
5:bne $s1, $s0, 4
6:
7: beq $s1, $s0, 10
8:
9: sub $s6, $s0, $s1
10: sub $s1, $s1, $s0
```

Output:

```
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 0, s2 : 0,
s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp :
0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 10, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 10, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 11, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 11, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 12, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 12, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 13, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]
```

```
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 13, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 14, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 14, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 15, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 15, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 15, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 15, s2 :
1, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 15, s1 : 0, s2 : 1,
s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp :
0, sp : 0, fp : 0, ra : 0]

Clock Cycles: 15
Number of times instructions in line 1, executed is 1
Number of times instructions in line 2, executed is 1
Number of times instructions in line 3, executed is 1
Number of times instructions in line 4, executed is 5
Number of times instructions in line 5, executed is 5
Number of times instructions in line 7, executed is 1
Number of times instructions in line 8, executed is 0
Number of times instructions in line 9, executed is 0
Number of times instructions in line 10, executed is 1
```

4) addi, sw, lw

```
Input:
addi $a0, $zero, 10
addi $t0, $zero, 100
sw $a0, 40($t0)
addi $t1, $zero, 104
lw $t2, 36($t1)

Output:
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 10, a1 : 0, a2 : 0, a3 : 0, t0 : 0,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 10, a1 : 0, a2 : 0, a3 : 0, t0 : 100,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 10, a1 : 0, a2 : 0, a3 : 0, t0 : 100,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 10, a1 : 0, a2 : 0, a3 : 0, t0 : 100,
t1 : 104, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0,
s2 : 0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 :
0, gp : 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 10, a1 : 0, a2 : 0, a3 : 0, t0 : 100,
t1 : 104, t2 : 10, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2
: 0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0,
gp : 0, sp : 0, fp : 0, ra : 0]

Clock Cycles: 5
Number of times instructions in line 1, executed is 1
Number of times instructions in line 2, executed is 1
Number of times instructions in line 3, executed is 1
Number of times instructions in line 4, executed is 1
Number of times instructions in line 5, executed is 1
```

Corner Case – 1 (Syntax errors):

1) Invalid parameters for addi:

```
Input:
1: addi $t0, $zero, 10
2:
3: addi $t1, $t0, $t0

Output:
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 10,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 10,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

Invalid syntax in line:3
```

2) Invalid parameters for sub:

```
Input:
1: addi $t0, $zero, 13
2: sub $t0, 5, 3

Output:
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 13,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

Missing Register(s) in the instruction in line: 2
```

3) Invalid parameters for lw:

```
Input:
1: lw $t0, 100

Output:
Invalid syntax in line:1
```

Note: For all possible syntax errors, our code flags an error.

Corner Case – 2 (Invalid Memory Address):

1) Memory address greater than 2^20:

```
Input:
1: addi $t0, $zero, 2000000000
2: lw $s0, 1000000($t0)

Output:
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 :
2000000000, t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0,
s1 : 0, s2 : 0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 :
0, k1 : 0, gp : 0, sp : 0, fp : 0, ra : 0]

Memory address out of bounds, line no: 2
```

2) Memory address lies in Instruction memory:

```
Input:
1: addi $t0, $zero, 0
2: lw $s0, 0($t0)

Output:
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 0, t1
: 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 : 0,
s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp :
0, sp : 0, fp : 0, ra : 0]

Invalid memory address, line no: 2
```

3) Memory address not a multiple of 4 (We assumed memory as chunks of 4 bytes):

```
Input:
1: addi $t0, $zero, 10
2: lw $s0, 107($t0)

Output:
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 10,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

Invalid memory address, line no: 2
```

Corner Case – 3 (Invalid Line Number):

1) Invalid line number in jump:

```
Input:
1: addi $t0, $zero, 10
2: j -100

Output:
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 10,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

Can't go to the mentioned line from jump, line 2
```

2) Invalid line number in branch:

```
Input:
1: addi $t0, $zero, 10
2: addi $t1, $zero, 12
3: bne $t0, $t1, 0

Output:
[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 10,
t1 : 0, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

[zero : 0, at : 0, v0 : 0, v1 : 0, a0 : 0, a1 : 0, a2 : 0, a3 : 0, t0 : 10,
t1 : 12, t2 : 0, t3 : 0, t4 : 0, t5 : 0, t6 : 0, t7 : 0, s0 : 0, s1 : 0, s2 :
0, s3 : 0, s4 : 0, s5 : 0, s6 : 0, s7 : 0, t8 : 0, t9 : 0, k0 : 0, k1 : 0, gp
: 0, sp : 0, fp : 0, ra : 0]

Can't go to the mentioned line from the branch, line 3
```

Corner Case – 4 (Invalid Command and Registers):

1) Invalid Command “li”:

```
Input:
1: li $v0, 4

Output:
Invalid instruction: li in line 1
```

2) Invalid Register:

```
Input:
1: addi $abc, $zero, 15

Output:
Invalid Syntax in line: 1
```

Corner Case – 5 (Modifying zero register):

1) Modifying zero register

```
Input:
1: addi $zero, $zero, 15

Output:
Zero register can't be changed, line: 1
```

Result: Output Values matched with the calculated values in all the cases.