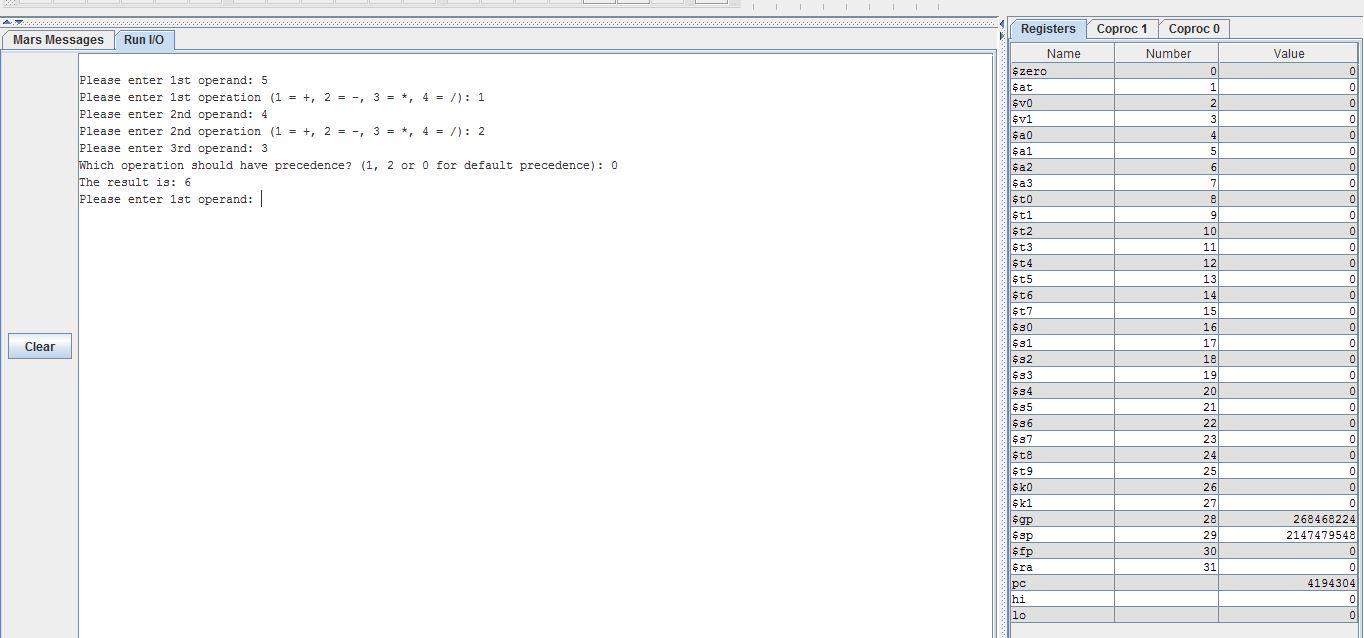
**COMPUTER ORGANISATION AND ARCHITECTURE REPORT**

1. A calculator MIPS code is used in this assignment.

Mars MIPS Simulator is used to simulate the above calculator code.

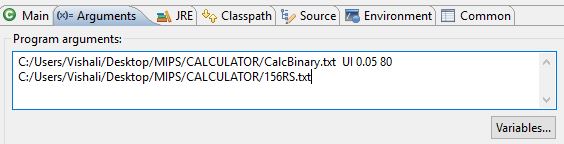
The code takes 3 operands and 2 operators of any choice from (+, -, \*, /) between them. Operator precedence can be specified by the user else default precedence is considered.

1. Normal code output without any UI Injected.

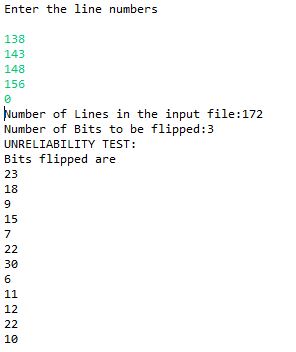


1. ERROR INJECTION RATE = 0.05%

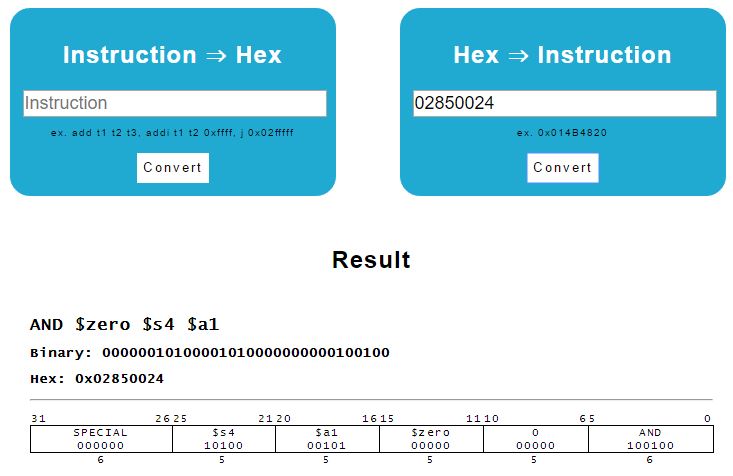
In our UI injection program, the error rate can be passed as an argument. The clustered UI rate can also be passed as an argument.



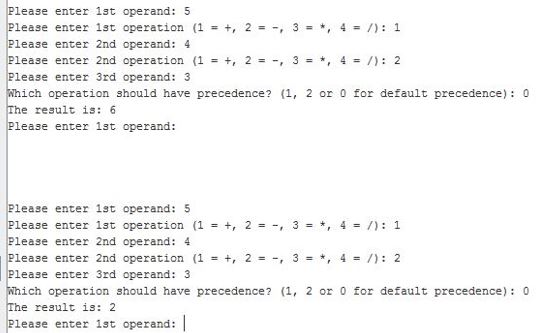
1. A simulator in java is written to inject UI’s in the MIPS code which takes the above arguments and the modified binary file is returned as output.

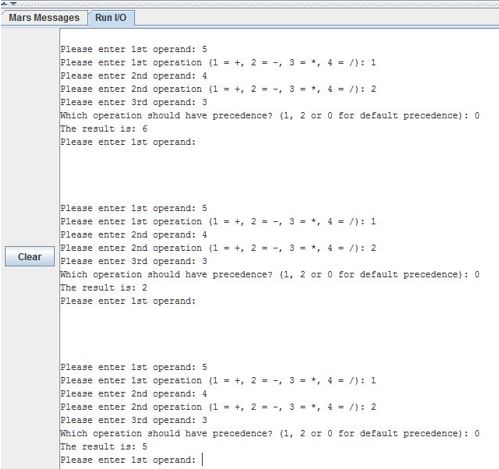
****

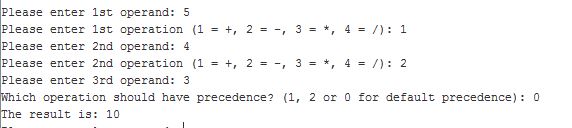
1. A java program is used to convert the binary file to hexadecimal file.
2. The hexadecimal instructions are converted back to MIPS instructions using online converter.

****

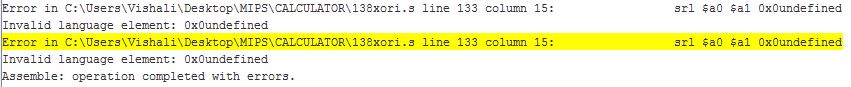
1. **TEST CASES FOR UNRELIABILITY  
   (WITH OUTPUT)**

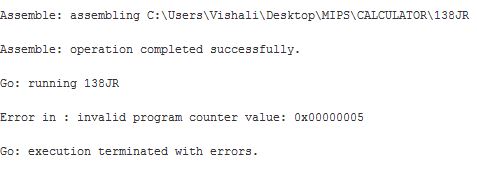
****

****

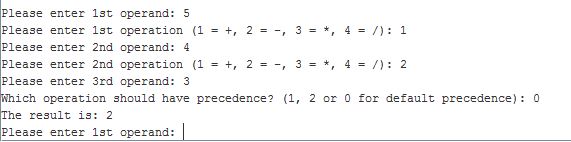
****

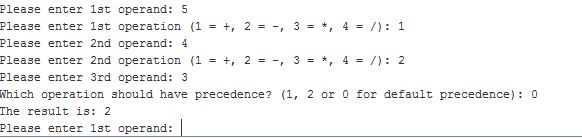
**TEST CASES FOR UNRELIABILITY  
(WITH ERRORS)**

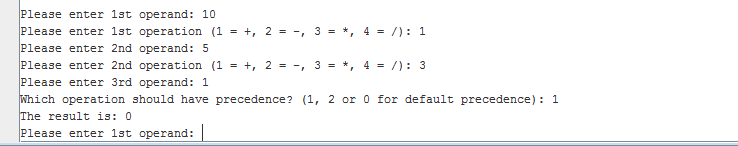
****

****

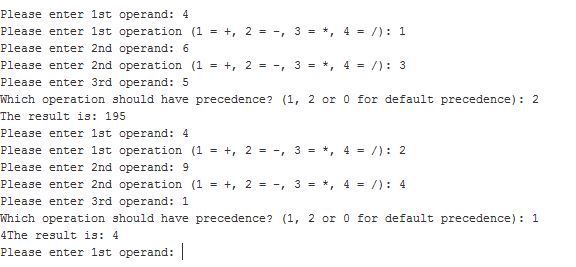
**TEST CASES FOR INSECURITY  
(WITH OUTPUT)**

****

****

****

**CLUSTERED UI RATE SAMPLE TEST CASES OUTPUT WITH 80/20**

****

1. **STATE PROBABILITIES**

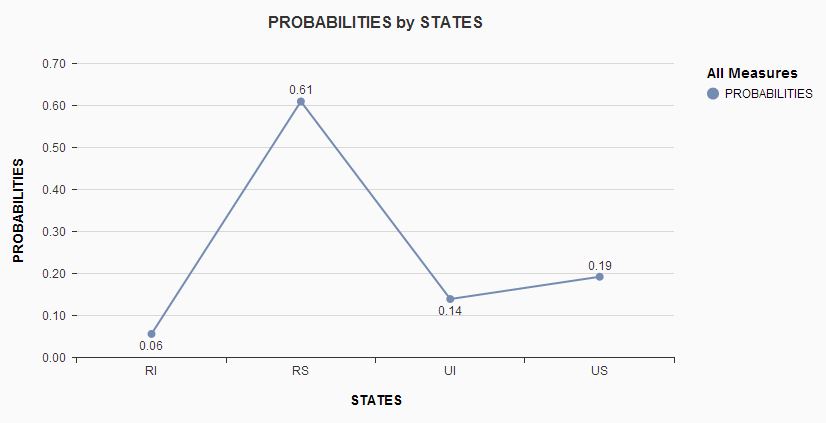
Probability of every state is calculated by considering the number of test cases passed / Total number of test cases taken.

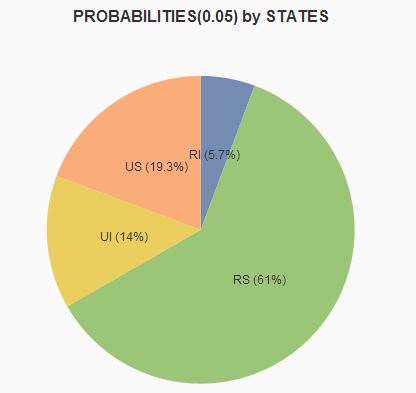
Probability of RS state = 183/300

Probability of RI state = 17/300

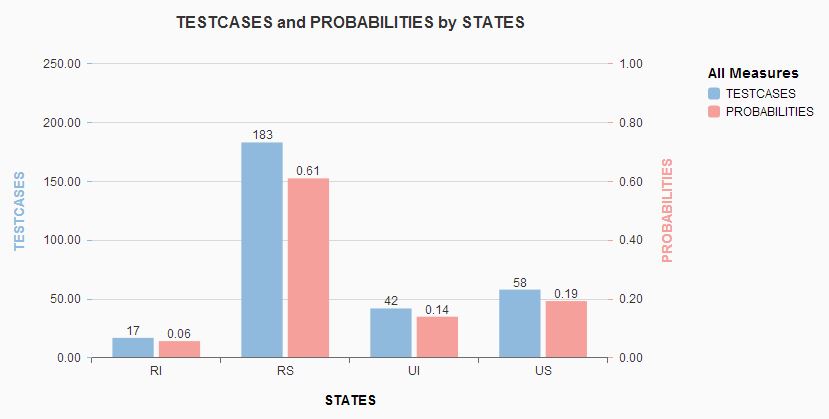
Probability of UI state = 42/300

Probability of US state = 58/300

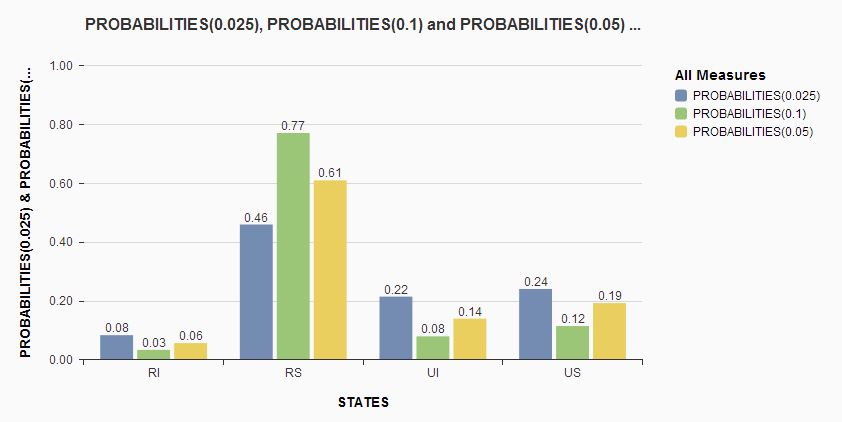
****

****

**TEST CASES WITH STATES PROBABILITY**

****

**STATE PROBABILITIES WITH VARIOUS ERROR RATES**

****

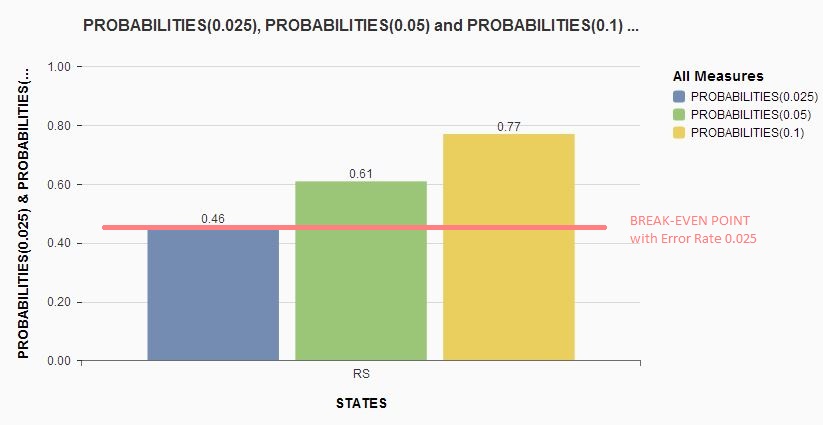
1. **ANALYTICAL STUDY(TAKING TASK PERFORMANCE INTO CONSIDERATION)**

Task Performance =

=

* Instruction count can increase or decrease based on the error rate. If the instruction count goes to infinity, the task completely fails.
* If injected error changes the type of instruction [R->J] then the instruction count might increase in one case and if there is no change in the type, it remains the same.
* Instruction count can vary with the same error rate depending upon the changes that took place after flipping.

**BREAK-EVEN POINT**

****

* Break-Even Point achieved considering 0.025% Error Injection is 0.46.
* The process would be reliable and secured with a minimum 0.46 probability.
* Break-Even varies with error injection. In our case 0.025% is the 1 bit flip occurrence.
* As the error rate increases, process halts by returning errors.

**Theoretical Probability**

* Probability varies with number of instructions and the error rate injection.
* As the Number of instructions increases with the same error rate the probability changes. Let us say we have a MIPS code with 100 instructions and a MIPS code with 100k instructions, error injection 0.01% is negligible in first case.
* The theoretical probability purely depends on the process considered.
* Theoretical Probability α (Number of Instructions)\*(Error Rate)