## Instruction Tested

### R Type Instruction:

```
24'b000_0001_0001_0001_0000_00000; //ADD $1,$1,$1

24'b000_0001_0011_0001_0001_00000; //SUB $1,$3,$1

24'b000_0010_0010_0010_0000_00000; //ADD $2,$2,$2

24'b000_0010_0001_0011_0000_00000; //ADD $3,$1,$2

24'b000_0010_0001_0011_0010_00000; //MUL $3,$1,$2  Result: 2

24'b000_0010_0100_0100_0011_00000; //DIV $4,$4,$2 Result: 2

24'b000_xxxx_0101_0101_0100_00000; //INC $5,$5 Result: 6

24'b000_xxxx_0110_0110_0101_00000; //DEC $6,$6 Result: 5

24'b000_0111_0111_0111_0110_00000; //AND $7,$7,$7 Result:7

24'b000_1001_1000_1000_0111_00000; //OR $8,$8,$9 Result:9

24'b000_1010_1001_1001_1000_00000; //XOR $9,$9,$10 Result: 3

24'b000_xxxx_1010_1010_1001_00000; //NOT $10,$10 Result:1111_1111_1111_0101
```

### Encryption Decryption:

```
24'b000_1010_xxxx_0101_1011_00000; //enc $5,$10

24'b000_0101_xxxx_1111_1100_00000; //dec $15,$5
```

### I Type (Store and Load Variant)

```
24'b001_0000_0101_1010_0001_00000; //STb 10,$5

24'b001_0000_0101_1010_0101_00000; //ST 10,$5

24'b001_0000_1111_0101_0000_00000; //LDbu $5,15

24'b001_0000_1111_0101_0010_00000; //LDb $5,15

24'b001_0000_1111_0101_0100_00000; //LD $5,15
```

### B Type (Conditional Branch)

```
24'b011_0100_0100_0000_1001_00000; //BNE $4,$4,9

24'b010_0100_0100_0000_0110_00000; //BEQ $4,$4,6

24'b010_0100_0100_0000_0011_00000; //BEQ $4,$4,3
```

### J Type (Unconditional Jump, Call, Retrieve)

```
24'b100_0000_0000_0001_1000_00000; //JMP 6

24'b100_0000_0000_0001_1001_00000; //CALL 6

24'b100_0000_0000_0011_0001_00000; //CALL 12

24'b100_xxxx_xxxx_xxxx_xx10_00000; //RET
```

### Fibonacci Series:

```
24'b000_0000_0000_0000_0000_00000; //ADD $0,$0,$0

24'b000_0001_0000_0001_0000_00000; //ADD $1,$0,$1

24'b000_0000_0001_0010_0000_00000; //ADD $2,$1,$0

24'b000_0001_0010_0011_0000_00000; //ADD $3,$2,$1

24'b000_0010_0011_0100_0000_00000; //ADD $4,$3,$2

24'b000_0011_0100_0101_0000_00000; //ADD $5,$4,$3

24'b000_0100_0101_0110_0000_00000; //ADD $6,$5,$4
```