I S A

BY CHUTIMUN 5917503

ISA SIMULATION CPU

- 24–bits
- Written in Java

Opcode 5 bits	Operand 1 3 bits	Binary number 16 bits

REGISTERS

```
private static String[] r = { "000", "001", "010", "011", "100", "101", "110", "111" };
```

THE BINARY OF OPCODE

```
private static String mov = "00001";
private static String add = "00010";
private static String sub = "00011";
private static String mul = "00100";
private static String div = "00101";
```

THE CLOCK CYCLE OF OPCODE

```
private static int c_mov = 1;
private static int c_add = 2;
private static int c_sub = 2;
private static int c_mul = 4;
private static int c_div = 4;
```

MOV

Operand 1 and Operand 2

mov r1 r2 -> to move the register 2 into register 1

Operand 1 and Value

mov r1 10 -> to move the value 10 into register 1

ADD

Operand 1 and Operand 2

add r1 r2 -> to add the register 2 into register 1

Operand 1 and Value

add r1 10 -> to add the value 10 into register 1

SUB

Operand 1 and Operand 2

sub r1 r2 -> to sub the register 2 into register 1

Operand 1 and Value

sub r1 10 -> to sub the value 10 into register 1

MUL

Operand 1 and Operand 2

mul r1 r2 0> to multiply the register 2 into register 1

Operand 1 and Value

mul r1 10 -> to multiply the value 10 into register 1

DIV

Operand 1 and Operand 2

div r1 r2 -> To divide the register 2 into register 1

Operand 1 and Value

div r1 10 -> to divide the value 10 into register 1

```
input order
Opcode: mov , add , sub , mul , div
Operand 1 : R0-R7
Operand 2 : R0-R7 or decimal value
-> end 0 0 <- is the opcode to stop input order
mov r0 6
mov r1 5
mov r2 r1
add r3 7
add r4 r1
sub r1 2
sub r0 r1
mul r4 6
mul r2 r1
div r3 3
div r4 r0
end 0 0
```

```
PC   Decoded          Encoded instructions(24-bit):                Clock cycles
P[0]mov r0,6         [00001  000  0000000000000110]                  1
P[1]mov r1,5         [00001  001  0000000000000101]                  1
P[2]mov r2,r1         [00001  010  0000000000000001]                  1
P[3]add r3,7         [00010  011  0000000000000111]                  2
P[4]add r4,r1         [00010  100  0000000000000001]                  2
P[5]sub r1,2         [00011  001  0000000000000010]                  2
P[6]sub r0,r1         [00011  000  0000000000000001]                  2
P[7]mul r4,6         [00100  100  0000000000000110]                  4
P[8]mul r2,r1         [00100  010  0000000000000001]                  4
P[9]div r3,3         [00101  011  0000000000000011]                  4
P[10]div r4,r0          [00101  100  0000000000000000]                  4
```

```
Steps of Register:
r0 = 6   [0000000000000110]
r1 = 5   [0000000000000101]
r2 = r1   [0000000000000001]
r3 = 7   [0000000000000111]
r4 = r1   [0000000000000001]
r1 = 2   [0000000000000010]
r0 = r1   [0000000000000001]
r4 = 6   [0000000000000110]
r2 = r1   [0000000000000001]
r3 = 3   [0000000000000011]
r4 = r0   [0000000000000000]
```

CPI OF THE PROGRAM

CPI 2.4545454545454546

SAMPLE CODE

```
if (number.charAt(i) >= '0' && number.charAt(i) <= '9') { // finding the number entered by user
int n = Integer.parseInt(number.substring(i, i + 1)); // calculating the actual number entered by
// the user

if (name.equals("mov") && second.equals(index(second)) // performing the move operation using two
// numbers
&& n == Integer.parseInt(binary(number), 2)) {

String m1 = "P[" + first(count) + "]" + (name + " " + second + "," + number) + "          ["
+ temp.getMov() + "   " + method(second) + "   " + binary(number) + "]"
+ "                    " + temp.getC_mov();

user.add(m1);
backup.push(second);
stack.push(number);
int d = temp.getC_mov();
count2++;
s = d * count2;
check.add(second);
store.add(second + "[" + temp.getMov() + "   " + method(second) + "   " + binary(number) + "]");

break;
```

```
} else if (name.equals("add") && second.equals(index(second))// performing add operations
&& n == Integer.parseInt(binary(number), 2)) {

String a1 = "P[" + first(count) + "]" + (name + " " + second + "," + number) + "          ["
+ temp.getAdd() + "   " + method(second) + "   " + binary(number) + "]"
+ "                    " + temp.getC_add();

user.add(a1);
int e = temp.getC_add();
count3++;
q = e * count3;
check.add(second);
store.add(second + "[" + temp.getAdd() + "   " + method(second) + "   " + binary(number) + "]");

break;
```

```
} else if (name.equals("mul") && second.equals(index(second)) // performing multiplication operation
&& n == Integer.parseInt(binary(number), 2)) {

String mul1 = "P[" + first(count) + "]" + (name + " " + second + "," + number) + "          ["
+ temp.getMul() + "   " + method(second) + "   " + binary(number) + "]"
+ "                      " + temp.getC_mul();
user.add(mul1);
int f = temp.getC_mul();
count4++;
j = f * count4;
check.add(second);
store.add(second + "[" + temp.getMul() + "   " + method(second) + "   " + binary(number) + "]");

break;
```

```
} else if (name.equals("sub") && second.equals(index(second))// performing subtract operation
&& n == Integer.parseInt(binary(number), 2)) {

String s1 = "P[" + first(count) + "]" + (name + " " + second + "," + number) + "          ["
+ temp.getSub() + "   " + method(second) + "   " + binary(number) + "]"
+ "                    " + temp.getC_sub();
user.add(s1);
int g = temp.getC_sub();// calling the function fro ISASimulation class
count5++;
z = g * count5;
check.add(second);
store.add(second + "[" + temp.getSub() + "   " + method(second) + "   " + binary(number) + "]");

break;
```