ISA Simulator

Pathompong Kunapermsiri 571-6712

About my ISA Simulator

- 24-bit ISA simulator program
- Written in Java 8
- There are 3 types of register
  - Register (r1-r9)
  - Upper register (m1-m9)
  - Lower register (d1-d9)
- There are 3 types of instruction
  - R-Type
  - I-Type
  - J-Type

Op-code and CPI

```
case "mov":
    if (ist.mov(w)) {
        System.out.println(" opcode: 000000 CPI: 1.0");
        cpi = cpi + 1;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
case "add":
    if (ist.add(w)) {
        System.out.println(" opcode 000001 CPI: 1.3");
        cpi = cpi + 1.3;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
```

```
case "sub":
    if (ist.sub(w)) {
        System.out.println(" opcode: 000011 CPI: 1.3");
        cpi = cpi + 1.3;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
case "mul":
    if (ist.mul(w)) {
        System.out.println(" opcode: 000111 CPI: 1.8");
        cpi = cpi + 1.8;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
```

Op-code and CPI

```
case "div":
    if (ist.div(w)) {
        System.out.println(" opcode: 001111 CPI: 1.8");
        cpi = cpi + 1.8;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
case "cop":
    if (ist.cop(w)) {
        System.out.println(" opcode: 011111 CPI: 1.0");
        cpi = cpi + 1;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
```

```
case "show":
    if(ist.show(w)) {
        System.out.println(" opcode: 111111 CPI: 1.9");
        cpi = cpi + 0.9;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
case "del":
    if(ist.del(w)) {
        System.out.println(" opcode: 111110 CPI: 1.0");
        cpi = cpi + 1;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
```

Op-Code and CPI

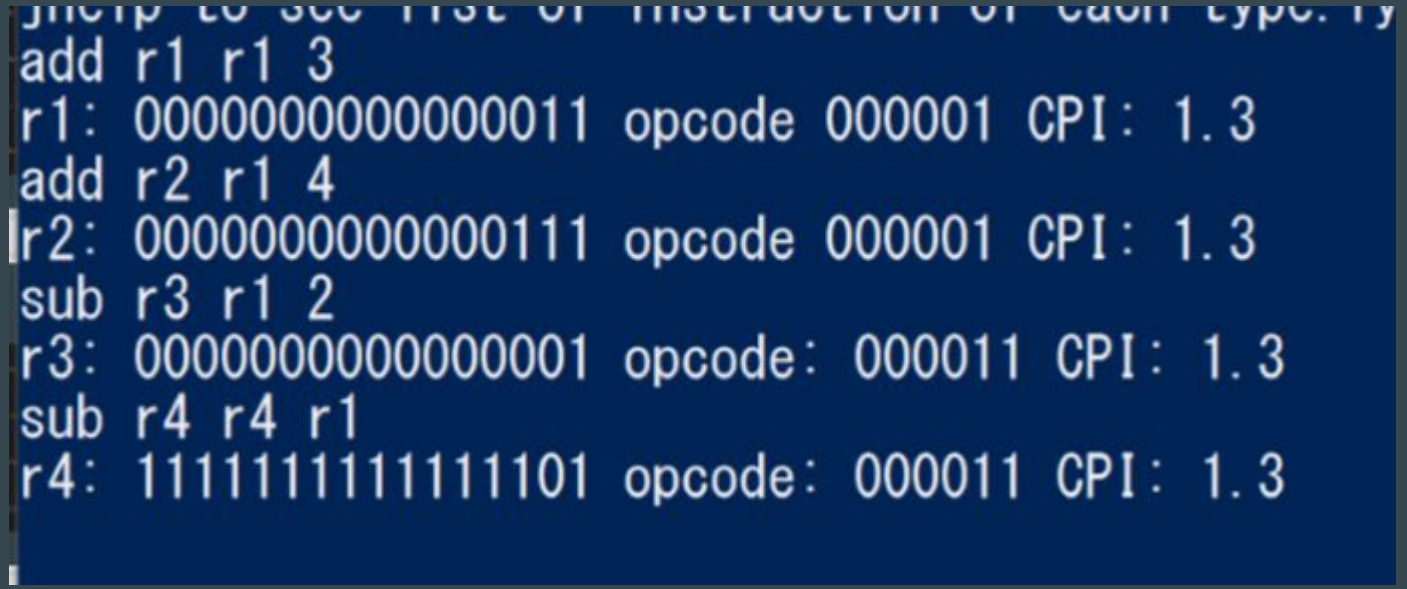
```
case "clear":
    if(ist.clear(w)) {
        System.out.println("opcode: 100100 CPI: 3");
        cpi = cpi + 3;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
```

```
end
Programe is terminated. CPI = 1.3666666666666667
Thank you
PS C:¥Users¥Top¥Documents¥ABAC¥isaSimulator>
```

R-Type Instruction

- Consist of 1 operant and 2 operator/value
- Add
  - Format: add [operant] [value/operator] [value/operator]
  - Detail: add 2 register or value then store in operant
- Sub
  - Format: sub [operant] [value/operator] [value/operator]
  - Detail: subtract 2 register or value then store in operant

R-Type Instruction



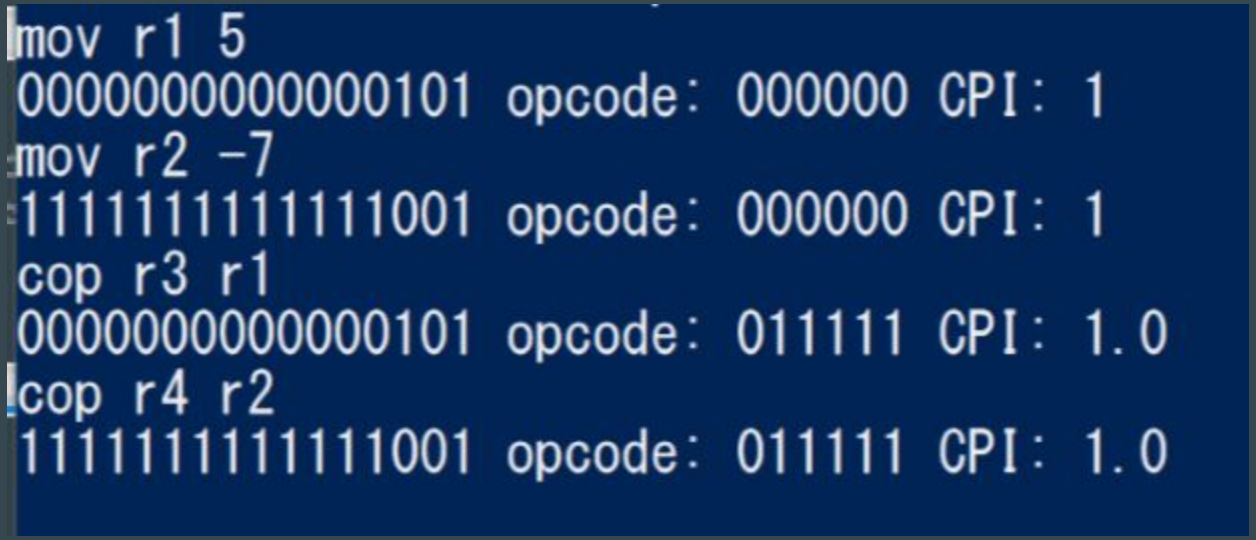
I-Type Instruction

- Consist of 1 operant and 1 operator
- Mov
  - Format: mov [operant] [value/operator]
  - Detail: move value or register to operant
- Cop
  - Format: cop [operant] [value/operator]
  - Detail: copy value or register to operant

I-Type Instruction

- Mul
  - Format: mul [operant] [value/operator]
  - Detail: multiply operant and value or register then store in operant
- Div
  - Format: div [operant] [value/operator]
  - Detail: divide operant and value or register then store in operant

I-Type Instruction



J-Type Instruction

- Consist 1 operant
- Del
  - Format: del [operant]
  - Detail: clear value in that register
- Show
  - Format: show [operant]
  - Detail: print that value

J-Type Instruction

```
mov r1 5
0000000000000101 opcode: 000000 CPI: 1
mov r2 -7
1111111111111001 opcode: 000000 CPI: 1
cop r3 r1
0000000000000101 opcode: 011111 CPI: 1.0
cop r4 r2
1111111111111001 opcode: 011111 CPI: 1.0
```

```
mul r3 r1
r3: 0000000000011001 m3: 0000000000000000 opcode: 000111 CPI: 1.8
mul r2 r3
r2: 1111111101010001 m2: 0000000000000000 opcode: 000111 CPI: 1.8
div r1 2
r1: 0000000000000010 d1: 0000000000000001 opcode: 001111 CPI: 1.8
div r2 4
r2: 1111111111010101 d2: 1111111111111101 opcode: 001111 CPI: 1.8
```

Source Code

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        System.out.println("Welcome to 24-bit ISA Simulator\n" + "========================\n"
                + "This simlator can handle any 16-bit or 8-bit Arithmetic operation."
                + "There are 9 16-bit registor (r1-r9), 9 registor for store remainder of division (d1-d9), "
                + "and 9 registoer for store upper result(m1-m9).\n"
                + "Type the opcode and follow by operant with space between them."
                + "For example, 'mov r0 23' or 'add r1 r2 r3. Any reminder or upper resault will be store at register "
                + "\nwhich their index is same as that register. For example, reminder of r3 will be store in d3"
                + "Please type rhelp, ihelp, or jhelp to see list of instruction of each type."
                + "Type 'end' to end programe and 'clear' to clear data in all register");

        Instruction ist = new Instruction();
        Scanner sc = new Scanner(System.in);

        String w = "";
        String a = "";
        double cpi = 0;
        int counter = 0;
        w = sc.nextLine();
        a = ist.checkCommand(w);

        while (!a.equals("end")) {

            switch (a) {
            case "mov":
                if (ist.mov(w)) {
                    System.out.println(" opcode: 000000 CPI: 1.0");
                    cpi = cpi + 1;
                    counter++;
                    break;
                } else {
                    System.out.println("Wrong input formation.");
                    break;
                }
            case "add":
                if (ist.add(w)) {
```

Source Code

```
case "sub":
    if (ist.sub(w)) {
        System.out.println(" opcode: 000011 CPI: 1.3");
        cpi = cpi + 1.3;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
case "mul":
    if (ist.mul(w)) {
        System.out.println(" opcode: 000111 CPI: 1.8");
        cpi = cpi + 1.8;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
case "div":
    if (ist.div(w)) {
        System.out.println(" opcode: 001111 CPI: 1.8");
        cpi = cpi + 1.8;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
    }
case "cop":
    if (ist.cop(w)) {
        System.out.println(" opcode: 011111 CPI: 1.0");
        cpi = cpi + 1;
        counter++;
        break;
    } else {
        System.out.println("Wrong input formation.");
        break;
```

Source Code

```
        }
    case "clear":
        if(ist.clear(w)) {
            System.out.println("opcode: 100100 CPI: 3");
            cpi = cpi + 3;
            counter++;
            break;
        } else {
            System.out.println("Wrong input formation.");
            break;
        }
    case "rhelp":
        System.out.println("1) add => add 2 register or value then store in operator register. Upper vaule will be store in 'm' register\
                + "2) sub => subtract 2 register or value then store in operator register. Below value will be store in 'd register.");
        break;
    case "ihelp":
        System.out.println("1) mov => move value or value from operant register to operator register\n"
                + "2) cop => copy value/value from operant register to operator register\n"
                + "3) mul => multiply operator register and operant/value then store in operator register. Any upper value will be store
                + "4) div => divide operator register and operant/value then store in operator register. Any reminder will be store in 'm
        break;
    case "jhelp":
        System.out.println("1) del => clear value in that register out\n"
                + "2) show => print that value");
        break;

    default:
        System.out.println("Wrong command.");
    }
    w = sc.nextLine();
    a = ist.checkCommand(w);

}
System.out.println("Programe is terminated. CPI = " + (cpi/counter) + "\nThank you");
sc.close();
```

Source Code

```java
public class Instruction {
    private int[] regStore = new int[9];
    private int[] remStore = new int[9];
    private int[] mulStore = new int[9];

    public Instruction() {
        for (int i = 0; i < 9; i++) {
            regStore[i] = 0;
            remStore[i] = 0;
            mulStore[i] = 0;
        }
    }

    // get value
    public int getReg(int index) {
        return regStore[index];
    }

    public int getRem(int index) {
        return remStore[index];
    }

    public int getMul(int index) {
        return mulStore[index];
    }

    // set value
    public void setReg(int index, int data) {
        regStore[index] = data;
    }

    public void setMul(int index, int data) {
        mulStore[index] = data;
    }

    public void setRem(int index, int data) {
        remStore[index] = data;
    }
```

```java
// check command
public String checkCommand(String s) {
    String[] temp = s.split(" ");
    return temp[0];
}

// get register index
public int GetregisterIndex(String s) {
    String string = "" + s.charAt(1);
    return Integer.parseInt(string);
}

// check register
public boolean CheckregisterInt(String s) {
    int d = 0;
    try {
        d = Integer.parseInt(s);
    } catch (NumberFormatException nfe) {
        return false;
    }
    if (d > 65535)
        return false;

    return true;
}

public char GetregisterType(String s) {
    return s.charAt(0);
}

// check register name
public boolean CheckregisterName(String s) {
    if (s.charAt(1) >= 48 && s.charAt(1) <= 57)
        if (s.charAt(0) == 'r' || s.charAt(0) == 'd' || s.charAt(0) == 'm')
            return true;

    return false;
}
```

Source Code

```
public int getRegister(int index, char name) {
    int register = 0;

    switch (name) {
    case 'r':
        register = regStore[index];
        break;
    case 'd':
        register = remStore[index];
        break;
    case 'm':
        register = mulStore[index];
        break;
    default:
        return -1;
    }

    return register;
}

public boolean setRegister(int index, char name, int data) {

    switch (name) {
    case 'r':
        regStore[index] = data;
        break;
    case 'd':
        remStore[index] = data;
        break;
    case 'm':
        regStore[index] = data;
        break;
    default:
        return false;
    }

    return true;
}
```

Source Code

```
public boolean clear(String s){
    if(s.equals("clear")){
        for(int i = 0; i < 9; i++){
            setReg(i, 0);
            setRem(i, 0);
            setMul(i, 0);
        }
        return true;
    } else
        return false;

}

// J-type
public boolean show(String s) {
    String[] temp = s.split(" ");

    if (temp.length == 2) {
        if (CheckregisterName(temp[1])) {

            char name = GetregisterType(temp[1]);
            int index = GetregisterIndex(temp[1]);
            System.out.print(name + "" + index + ": " + BinarytohexaString(getRegister(index, name)));
            return true;
        } else
            return false;

    } else
        return false;
}
```

Source Code

```
public boolean del(String s) {
    String[] temp = s.split(" ");

    if (temp.length == 2) {
        if (CheckregisterName(temp[1])) {

            char name = GetregisterType(temp[1]);
            int index = GetregisterIndex(temp[1]);
            setRegister(index, name, 0);
            System.out.print(BinarytohexaString(getRegister(index, name)));
            return true;
        } else
            return false;

    } else
        return false;
}

// R-type
public boolean add(String s) {
    String[] temp = s.split(" ");
    int operator1 = 0;
    int operator2 = 0;
    int index = 0;
    char name;
    boolean hi = false;

    if (temp.length == 4) {
        if (CheckregisterName(temp[1])) {

            if (CheckregisterInt(temp[2])) {
                operator1 = Integer.parseInt(temp[2]);

            } else if (CheckregisterName(temp[2])) {
                index = GetregisterIndex(temp[2]);
                name = GetregisterType(temp[2]);
                operator1 = getRegister(index, name);

            } else
```

Source Code

```
        index = GetregisterIndex(temp[2]);
        name = GetregisterType(temp[2]);
        operator1 = getRegister(index, name);

    } else

        return false;

    if (CheckregisterInt(temp[3])) {
        operator2 = Integer.parseInt(temp[3]);

    } else if (CheckregisterName(temp[3])) {
        index = GetregisterIndex(temp[3]);
        name = GetregisterType(temp[3]);
        operator2 = getRegister(index, name);

    } else
        return false;

    operator1 = operator1 + operator2;
    index = GetregisterIndex(temp[1]);
    name = GetregisterType(temp[1]);

    if (operator1 > 65535) {
        operator1 = operator1 - 65535;
        setMul(index, operator1);
        hi = true;
    }

    setReg(index, operator1);
    System.out.print("r" + index + ": " + BinarytohexaString(getRegister(index, name)));

    if (hi) {
        System.out.print(" m" + index + ": " + BinarytohexaString(getMul(index)));
    }

    return true;

- else
    return false;
```

Source Code

- See the rest in attached file

To run program

- Java -jar isa.jar