# NRegister.java

```java
// Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas

package jminusminus;

import java.util.ArrayList;
import java.util.Collections;

/**
 * An abstraction for a (virtual or physical) register.
 */

abstract class NRegister {

    /** Register number. */
    protected int number;

    /** Register name. */
    protected String name;

    /**
     * Construct an NRegister.
     * 
     * @param number
     *            register number.
     * @param name
     *            register name.
     */

    protected NRegister(int number, String name) {
        this.number = number;
        this.name = name;
    }

    /**
     * Return the number of this register.
     * 
     * @return register number.
     */

    public int number() {
        return number;
    }

    /**
     * Return the name of this register.
     * 
     * @return register name.
     */

    public String name() {
        return name;
    }

}

/**
 * An abstraction for a virtual register.
 */

class NVirtualRegister extends NRegister {

    /** Type (short name) of value in register. */
    private String sType;

    /** Type (long name) of value in register. */
    private String lType;

    /**
     * Construct an NVirutalRegister.
     *
     * @param number
     *            register number.
     * @param sType
     *            type (short name) of value in register.
     * @param lType
     *            type (long name) of value in register.
     */

    public NVirtualRegister(int number, String sType, String lType) {
        super(number, "V" + number);
        this.sType = sType;
        this.lType = lType;
    }

    /**
     * Return a string representation of this virtual register.
     *
     * @return string representation.
     */

    public String toString() {
        return "[" + name + "|" + sType + "]";
    }

}

/**
 * An abstraction for a physical (SPIM) register.
 */

class NPhysicalRegister extends NRegister {

    /**
     * Maximum number of physical registers used for allocation, starting at T0.
     */
    public static int MAX_COUNT = 8;

    // Constants identifying the physical registers. These
    // can be used as indices into the static regInfo array
    // to access the representations of the corresponding
    // registers.

    /** Constant 0. */
    public static final int ZERO = 0;

    /** Reserved for assembler. */
    public static final int AT = 1;

    /** Expression evaluation and results of a function. */
    public static final int V0 = 2;

    /** Expression evaluation and results of a function. */
    public static final int V1 = 3;

    /** Argument 1. */
    public static final int A0 = 4;

    /** Argument 2. */
    public static final int A1 = 5;

    /** Argument 3. */
    public static final int A2 = 6;

    /** Argument 4. */
    public static final int A3 = 7;
```

```java

    /** Temporary (not preserved across call). */
    public static final int T0 = 8;

    /** Temporary (not preserved across call). */
    public static final int T1 = 9;

    /** Temporary (not preserved across call). */
    public static final int T2 = 10;

    /** Temporary (not preserved across call). */
    public static final int T3 = 11;

    /** Temporary (not preserved across call). */
    public static final int T4 = 12;

    /** Temporary (not preserved across call). */
    public static final int T5 = 13;

    /** Temporary (not preserved across call). */
    public static final int T6 = 14;

    /** Temporary (not preserved across call). */
    public static final int T7 = 15;

    /** Temporary (preserved across call). */
    public static final int S0 = 16;

    /** Temporary (preserved across call). */
    public static final int S1 = 17;

    /** Temporary (preserved across call). */
    public static final int S2 = 18;

    /** Temporary (preserved across call). */
    public static final int S3 = 19;

    /** Temporary (preserved across call). */
    public static final int S4 = 20;

    /** Temporary (preserved across call). */
    public static final int S5 = 21;

    /** Temporary (preserved across call). */
    public static final int S6 = 22;

    /** Temporary (preserved across call). */
    public static final int S7 = 23;

    /** Temporary (not preserved across call). */
    public static final int T8 = 24;

    /** Temporary (not preserved across call). */
    public static final int T9 = 25;

    /** Reserved for OS kernel. */
    public static final int K0 = 26;

    /** Reserved for OS kernel. */
    public static final int K1 = 27;

    /** Pointer to global area. */
    public static final int GP = 28;

    /** Stack pointer. */
    public static final int SP = 29;

    /** Frame pointer. */
    public static final int FP = 30;

    /** Return address (used by function call). */
    public static final int RA = 31;

    /**
     * Maps register number to the register's representation.
     */
    public static final NPhysicalRegister[] regInfo = {
            new NPhysicalRegister(0, "zero"), new NPhysicalRegister(1, "at"),
            new NPhysicalRegister(2, "v0"), new NPhysicalRegister(3, "v1"),
            new NPhysicalRegister(4, "a0"), new NPhysicalRegister(5, "a1"),
            new NPhysicalRegister(6, "a2"), new NPhysicalRegister(7, "a3"),
            new NPhysicalRegister(8, "t0"), new NPhysicalRegister(9, "t1"),
            new NPhysicalRegister(10, "t2"), new NPhysicalRegister(11, "t3"),
            new NPhysicalRegister(12, "t4"), new NPhysicalRegister(13, "t5"),
            new NPhysicalRegister(14, "t6"), new NPhysicalRegister(15, "t7"),
            new NPhysicalRegister(16, "s0"), new NPhysicalRegister(17, "s1"),
            new NPhysicalRegister(18, "s2"), new NPhysicalRegister(19, "s3"),
            new NPhysicalRegister(20, "s4"), new NPhysicalRegister(21, "s5"),
            new NPhysicalRegister(22, "s6"), new NPhysicalRegister(23, "s7"),
            new NPhysicalRegister(24, "t8"), new NPhysicalRegister(25, "t9"),
            new NPhysicalRegister(26, "k0"), new NPhysicalRegister(27, "k1"),
            new NPhysicalRegister(28, "gp"), new NPhysicalRegister(29, "sp"),
            new NPhysicalRegister(30, "fp"), new NPhysicalRegister(31, "ra"), };

    /**
     * Construct an NPhysicalRegister.
     * 
     * @param number
     *            number of the register.
     * @param name
     *            name of the register.
     */

    public NPhysicalRegister(int number, String name) {
        super(number, name);
    }

    /**
     * Return a string representation of this physical register.
     * 
     * @return string representation.
     */

    public String toString() {
        return "$" + name();
    }

}

```