```
#include <iostream>
#include <stdio.h>
#include <string>
#include <cstdlib>
#include <cassert>

#include "VirtualMachine.h"

using namespace std;

int VirtualMachine::get_clock()
{
    return clock;
}

void VirtualMachine::run(fstream& objectCode, fstream& in, fstream& out)
{
    const int debug = false;
    int opcode, rd, i, rs, constant, addr, j;

    base = 0;
    for (limit = 0; objectCode >> mem[limit]; limit++);

    sr = 2;
    sp = msize;
    pc = 0;
    while (pc < limit) {
        ir = mem[pc];
        pc++;
        opcode = (ir&0xf800)>>11;
        rd = (ir&0x600)>>9;
        i = (ir&0x100)>>8;
        rs = (ir&0xc0)>>6;
        addr = ir&0xff;
        constant = addr;

        clock++;

        if (opcode == 0) { /* load loadi */
            if (i) {
                if (constant&0x80) constant |= 0xff00;
                r[rd] = constant;
            } else {
                if (addr >= limit) {
                    out << "Out of bound error.\n";
                    return;
                }
                r[rd] = mem[addr];
                clock += 3;
            }

        } else if (opcode == 1) { /* store */
            if (addr >= limit) {
                out << "Out of bound error.\n";
                return;
            }
            mem[addr] = r[rd];
            clock += 3;

        } else if (opcode == 2) { /* add addi */
            int sign1 = (r[rd]&0x8000)>>15;
            int sign2;
            if (i) {
                sign2 = (constant&0x80)>>7;
            // sign extend both operands to perform operation
            if (sign2) constant |= 0xffffff00;
            if (r[rd] & 0x8000) r[rd] |= 0xffff0000;
            r[rd] = r[rd] + constant;
        } else {
            sign2 = (r[rs]&0x8000)>>15;
            int temp = r[rs];
            // sign extend both operands to perform operation
            if (sign2) temp |= 0xffff0000;
            if (r[rd] & 0x8000) r[rd] |= 0xffff0000;
            r[rd] = r[rd] + temp;
        }

        // set CARRY
        if (r[rd]&0x10000) sr |= 0x1;
        else sr &= 0xfffe;

        // set OVERFLOW
        if (sign1 == sign2 and sign1 != (r[rd]&0x8000)>>15)
            sr |= 0x10;
        else
            sr &= 0xffef;

        // keep it at 16 bits
        r[rd] &= 0xffff;

    } else if (opcode == 3) { /* addc addci */
        int sign1 = (r[rd]&0x8000)>>15;
        int sign2;
        if (i) {
            sign2 = (constant&0x80)>>7;
            // sign extend both operands to perform operation
            if (sign2) constant |= 0xffffff00;
            if (r[rd] & 0x8000) r[rd] |= 0xffff0000;
            r[rd] = r[rd] + constant + sr&0x1;
        } else {
            sign2 = (r[rs]&0x8000)>>15;
            int temp = r[rs];
            // sign extend both operands to perform operation
            if (sign2) temp |= 0xffff0000;
            if (r[rd] & 0x8000) r[rd] |= 0xffff0000;
            r[rd] = r[rd] + temp + sr&0x1;
        }

        // set CARRY
        if (r[rd]&0x10000) sr |= 0x1;
        else sr &= 0xfffe;

        // set OVERFLOW
        if (sign1 == sign2 and sign1 != (r[rd]&0x8000)>>15)
            sr |= 0x10;
        else
            sr &= 0xffef;

        // keep it at 16 bits
        r[rd] &= 0xffff;

...
    } else {
        cout << "Bad opcode = " << opcode << endl;
        exit(3);
    }

    if (debug) {
        printf("ir=%d op=%d rd=%d i=%d rs=%d const=%d addr=%d\n", ir, opcode, rd, i, rs,
constant, addr);
            printf("r[0]=%d r[1]=%d r[2]=%d r[3]=%d pc=%d sr=%d sp=%d clock=%d\n\n", r[0], r[1],
r[2], r[3], pc, sr, sp, clock);
            //char c;
            //cin>>c;
        }
    }

    if (debug) {
        for (j=0; j<limit; j++) {
            printf("%8d", mem[j]);
            if ((j%8)==7) printf("\n");
        }
        cout << endl;
    }
} /* main */
```