Source codes

Binary simulator

Mohammed Alaa Elkomy

Backend

WWW.JS

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('express libraries:server');
var http = require('http');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}

/**
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

app.js

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');

var routes = require('./routes/router');
var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

// uncomment after placing your favicon in /public
//app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', routes);

app.use('/f', express.static('public/files'));

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handlers

/*
                    // development error handler
                    // will print stacktrace
                    if (app.get('env') === 'development') {
                      app.use(function(err, req, res, next) {
                        res.status(err.status || 500);
                        res.render('error', {
                          message: err.message,
                          error: err
                        });
                      });
                    }
 */
                    // production error handler
                    // no stacktraces leaked to user
                    app.use(function(err, req, res, next) {
                      res.status(err.status || 500);
                      res.render('error', {
                        message: err.message,
                        error: {}
                      });
                    });


module.exports = app;
```

router.js

```
var express = require('express');
var Comp = require('./../model/Computations');

var router = express.Router();

/* GET simulator page. */
router.get('/', function(req, res) {
  res.render('home');
});



router.post('/', function(req, res) {

  var params=JSON.parse(req.body.params);

  var operand1=Comp.toBitString32(params.Field1,params.Base);
  var operand2=Comp.toBitString32(params.Field2,params.Base);



  var Response={};
  switch (params.Operation){
    case 0: //Addition
      Response.Arith=Comp.Addition(operand1,operand2,params.Signed);
    break;

    case 1: //Subtraction
      Response.Arith=Comp.Subtraction(operand1,operand2,true);
    break;

    case 2: //Normal Mult
      Response.Arith=Comp.NormalMultplication(operand1,operand2,params.Signed);
      break;

    case 3: //Booth Mult
      Response.Arith=Comp.BoothMultplication(operand1,operand2,params.Signed);
      break;

    case 4: //Bit-pair Mult
      Response.Arith=Comp.BitPairMultplication(operand1,operand2,params.Signed);
      break;

    case 5: //Restoring division

      Response=Comp.RestoringDivision32Bit(operand1,operand2,params.Signed);
      break;

    case 6: //Non-restoring Division
      Response=Comp.NonRestoringDivision32Bit(operand1,operand2,params.Signed);
      break;
  }
  if(params.Operation <5)
  Response.Text=Comp.flagsF();

  res.send(Response);
});
```

```
module.exports = router;
```

Model

baseUnit.js

```
/**
 * Created by mohammed on 13/05/16.
 */

var ZeroString32='00000000000000000000000000000000';
var OneString32= '11111111111111111111111111111111';
var shortLine  = '---------------------------------------- ---------';
var longLine= '-----------------------------------------------------------------------
----------  -------------';
var LINE='-----------------------------------------'

var
PowersOfTwo=[1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768,65536,131072
,262144,524288,1048576,2097152,4194304,8388608,16777216,33554432,67108864,134217728,26
8435456,536870912,1073741824,2147483648,4294967296,8589934592,17179869184,34359738368,
68719476736,137438953472,274877906944,549755813888,1099511627776,2199023255552,4398046
511104,8796093022208,17592186044416,35184372088832,70368744177664,140737488355328,2814
74976710656,562949953421312,1125899906842624,2251799813685248,4503599627370496,9007199
254740992,18014398509481984,36028797018963970,72057594037927940,144115188075855870,288
230376151711740,576460752303423500,1152921504606847000,2305843009213694000,46116860184
27388000,9223372036854776000,18446744073709552000];

var REDUNDANT_BITS;



function CalculateRedendandBits(bitstring){
    var length= 0;
    for(var i =0 ; i< bitstring.length  ; i++)
    {
        if(bitstring.charAt(i)=='0'){
            length++;
        }
        else break;
    }
    return length;
}


function removeParenthesis(bitstring){
 return   bitstring.substring(2,bitstring.length-2);
}

function toDecimal(bitstring,Signed){
  if(bitstring.length ==66)bitstring=bitstring.substring(2,bitstring.length)

    var decimal=0;
    if(Signed && bitstring.charAt(0)=='1')
    {
        var val=toTwoS_Complemnt(bitstring);
        decimal=toDecimal(  val     ,false);
        decimal ='-'+removeParenthesis(decimal);

    }
    else

    for(var i =bitstring.length-1 ; i>=0 ; i-- ){

        decimal +=(Number(bitstring.charAt(i)))*PowersOfTwo[(bitstring.length-1-i)];
    }


    return ' ('+decimal+')\n';
}


function BitstringFormat(bitstring) {

    bitstring =(bitstring.substring(  REDUNDANT_BITS ,bitstring .length));


    for (var i = bitstring.length-1; i >0 ; i -= 4){
        bitstring=  replaceAt(bitstring,i,bitstring.charAt(i)+' ');
        }



    if( bitstring.length==83)
        bitstring=bitstring.substring(3,83);



    return bitstring;
}

function fullAdderLogic(x,y,Cin){
    var sum = x ^ y ^ Cin;
    var Cout = (x&y)|(y&Cin)|(x&Cin);
    return {sum :sum,
        carry:Cout}
}

function mltiAdder64Bit(listOfOperands,Signed){
    var ex='';

    if(listOfOperands[0].length !=64)
        ex ='00';

    var tempRes=ex+ZeroString32.concat(ZeroString32); //64bitString of zeros



    for(var i=0 ; i< listOfOperands.length; i++){
        tempRes = addNBits(tempRes, listOfOperands[i],Signed,false/*update flags at
the end*/);
    }
    return tempRes ;
}
```

```
function addNBits  (operand1,operand2,Signed,IsTWOS_OrDoNotUpdateFlags){
    if(!IsTWOS_OrDoNotUpdateFlags)
        resetFlags();

    var result;

    if(operand1 .length ==32) //normal addition and subtraction (32 bits)
        result=ZeroString32;

    else if (operand1 .length ==64)
        result=ZeroString32.concat(ZeroString32); //signed multiplication (64 bits)
    else if (operand1 .length ==66)
        result=ZeroString32.concat(ZeroString32)+'00';   //unsigned booth (66 bits)
    else
        result=ZeroString32+'0'; //for division (33 bits)


    var tempCarry=0;

    for(var i = operand1.length-1 ; i >-1 ; i--){
        var FullAdderOutput= fullAdderLogic(operand1[i],operand2[i],tempCarry);
        result =replaceAt(result,i,FullAdderOutput.sum);
        if( !IsTWOS_OrDoNotUpdateFlags
&&Signed&&(!i)&&(tempCarry^FullAdderOutput.carry) ){ FLAGS.OVERFLOW =true;  }
        tempCarry=FullAdderOutput.carry;
    }

    if(!IsTWOS_OrDoNotUpdateFlags)
         UpdateFlags(result,tempCarry);
    return  result;
}


function sub32Bit(operand1,operand2,Signed){
    return  addNBits  (operand1,  toTwoS_Complemnt(operand2),Signed,false);
}

function multiAddition(listOfOperands,Signed,EXTENDLINE){

    var ex='';

    if(listOfOperands[0].length !=64)
        ex ='00';

    var tempRes=ex+ZeroString32.concat(ZeroString32); //64bitString of zeros


    var outputString ='\n  '+BitstringFormat(listOfOperands[listOfOperands.length-
1])+toDecimal(listOfOperands[listOfOperands.length-1],Signed);
    tempRes = addNBits(tempRes, listOfOperands[listOfOperands.length-
1],Signed,false/*update flags at the end*/);


    for(var i=listOfOperands.length-2 ; i>=0 ; i--){
        tempRes = addNBits(tempRes, listOfOperands[i],Signed,false/*update flags at
the end*/);
        var binary =BitstringFormat(listOfOperands[i]);

        var NoOfSteps=binary.length-listOfOperands.length+i-1;
        for(var j=binary.length-1 ; j>NoOfSteps;j--){
            binary=replaceAt(binary,j,' ');
        }
        outputString += '+ '+binary+toDecimal(listOfOperands[i],Signed) ;
    }

    return  outputString+longLine.substring((5/4)*REDUNDANT_BITS +(EXTENDLINE?-4:0)
,longLine.length)+'\n  '+BitstringFormat( tempRes )+toDecimal(tempRes,Signed);
}

function getRedundantBitsForAddORSub(operand1,operand2,Signed){
    REDUNDANT_BITS= (Signed?-
2:0)+Math.min(CalculateRedendandBits(operand1),CalculateRedendandBits(operand2));
}

function Addition  (operand1,operand2,Signed){
    getRedundantBitsForAddORSub(operand1,operand2,Signed);
    var addRes=addNBits(operand1,  operand2,Signed,false);
    var outputString ='\n  '+BitstringFormat(operand1)+toDecimal(operand1,Signed)+'+
'+BitstringFormat(operand2)+toDecimal(operand2,Signed)+'-
'+shortLine.substring(5*REDUNDANT_BITS/4+1,shortLine.length)+'\n  '+BitstringFormat(
addRes )+toDecimal(addRes,Signed);
    return  outputString;
}



function Subtraction(operand1,operand2,Signed){
    getRedundantBitsForAddORSub(operand1,operand2,Signed);
    var outputString ='\n  '+BitstringFormat(operand1)+toDecimal(operand1,Signed)+'-
'+BitstringFormat(operand2)+toDecimal(operand2,Signed)+'-
'+shortLine.substring((Signed?-1:0)+5*REDUNDANT_BITS/4+1,shortLine.length)+'
'+Addition(operand1,toTwoS_Complemnt(operand2),Signed);
    return  outputString;
}

function flagsF(){
   return '  V: '+Number(FLAGS.OVERFLOW)+'\tC: '+Number(FLAGS.CARRY)+'\tN:
'+Number(FLAGS.NEGATIVE)+'\tZ: '+Number(FLAGS.ZERO)+'\n';

}



var FLAGS ={
    OVERFLOW :false,
    CARRY :false,
    NEGATIVE :false,
    ZERO :false,
}


function  resetFlags() {
    FLAGS = {
        OVERFLOW: false,
        CARRY: false,
        NEGATIVE: false,
        ZERO: false,
    }
}

function  UpdateFlags(Result,Carry) {

    if(Carry) FLAGS.CARRY =true;
    if(Result[0] =='1') FLAGS.NEGATIVE=true;
    if(Result == ZeroString32)  FLAGS.ZERO=true;
}
// replace the nth character of 's' with 't'
function replaceAt(s, n, t) {
    return s.substring(0, n) + t + s.substring(n + 1);
}

function toTwoS_Complemnt (value){

    var ONE;

    if(value .length ==32)
        ONE=ZeroString32.substring(0, 31).concat('1'); //normal addition and
subtraction (32 bits)

    else if (value .length ==64)
        ONE=(ZeroString32.concat(ZeroString32)).substring(0, 63).concat('1'); //signed
multiplication (64 bits)
    else if (value.length ==66)
        ONE='0'+(ZeroString32.concat(ZeroString32)).concat('1');  //unsigned booth (66
bits)
    else
        ONE=ZeroString32+'1'; //for division (33 bits)


    for(var i = 0 ; i<value.length; i ++)
    {
        var ch = value[i]=='0'? '1':'0';
        value =replaceAt(value,i,ch);
    }
    value = addNBits(value,ONE ,false,true);

    return value;
}

function toBitString32  (value, base)  {
    var BinaryEqu;
    switch (base){
        case 8:
            BinaryEqu =parseInt(value, 8).toString(2);
            break;
        case 10:
            BinaryEqu= parseInt(value, 10).toString(2);
            break;
        case 16:
            BinaryEqu=parseInt(value, 16).toString(2);
            break;
        default:
            BinaryEqu=value;
            break;
    }


    if(BinaryEqu[0] == '-'){
        BinaryEqu=ZeroString32.substring(0, 32-
BinaryEqu.length+1).concat(BinaryEqu.substring(1,BinaryEqu.length ));

        BinaryEqu =toTwoS_Complemnt(BinaryEqu);
    }
    else
        BinaryEqu=ZeroString32.substring(0, 32-BinaryEqu.length).concat(BinaryEqu);

    return BinaryEqu;
```

```
}

function shiftRight(operand,SI) {
    FLAGS.CARRY=operand.charAt(operand.length-1);

    return SI+operand.substring(0,operand.length-1);
}

function shiftLeft(operand,SI) {
    FLAGS.CARRY=operand.charAt(0);
    return operand.substring(1,operand.length)+SI;
}

function extendTo64Bit(value, signed){
    var ex0=''; var ex1='';
    if(value.length != 32)  {ex0='0';ex1='1';}
    var result;
    if(signed && value.charAt(0)=='1')
        result=ex1+OneString32.concat(value);
    else  result=ex0+ZeroString32.concat(value);

    return result;
}


module.exports = {
    OneString32:OneString32,
    ZeroString32:ZeroString32,
    toTwoS_Complemnt:toTwoS_Complemnt,
    toBitString32:toBitString32,
    extendTo64Bit:extendTo64Bit,
    shiftLeft:shiftLeft,
    shiftRight:shiftRight,
    UpdateFlags:UpdateFlags,
    resetFlags:resetFlags,
    replaceAt:replaceAt,
    flagsF :flagsF,
    sub32Bit:sub32Bit,
    addNBits:addNBits,
    Subtraction:Subtraction,
    Addition:Addition,
    multiAddition,multiAddition,
    mltiAdder64Bit:mltiAdder64Bit,
    BitstringFormat:BitstringFormat,
    shortLine:shortLine,
    LINE:LINE,
    GET_REDUNDANT_BITS: function () {
        return REDUNDANT_BITS;
    },
    SET_REDUNDANT_BITS: function (RB) {
        REDUNDANT_BITS=RB;
    },
    toDecimal:toDecimal,
    CalculateRedendandBits,CalculateRedendandBits

};
```

utilUnit.js

```
var baseUnit = require('./../model/baseUnit');
```

```
function invertBit(bit){
    if(bit=='0')
    return '1';
    else return '0';
}


module.exports = {
    OneString32:baseUnit.OneString32,
    ZeroString32:baseUnit.ZeroString32,
    toTwoS_Complemnt:baseUnit.toTwoS_Complemnt,
    toBitString32:baseUnit.toBitString32,
    extendTo64Bit:baseUnit.extendTo64Bit,
    shiftLeft:baseUnit.shiftLeft,
    shiftRight:baseUnit.shiftRight,
    replaceAt:baseUnit.replaceAt,
    flagsF :baseUnit.flagsF,
    invertBit:invertBit,
    toDecimal:baseUnit.toDecimal,
    BitstringFormat:baseUnit.BitstringFormat,
    CalculateRedendandBits:baseUnit.CalculateRedendandBits,
    shortLine:baseUnit.shortLine,
    LINE:baseUnit.LINE,
    GET_REDUNDANT_BITS: baseUnit.GET_REDUNDANT_BITS,
    SET_REDUNDANT_BITS: baseUnit.SET_REDUNDANT_BITS

};
```

addUnit.js

```
var baseUnit = require('./../model/baseUnit');

module.exports = {
    Subtraction:baseUnit.Subtraction,
    Addition:baseUnit.Addition,
    sub32Bit:baseUnit.sub32Bit,
    addNBits:baseUnit.addNBits,
    sub32BitF:baseUnit.sub32BitF,
    addNBitsF:baseUnit.addNBitsF,
    multiAddition:baseUnit.multiAddition
};
```

mulUnit.js

```
/**
 * Created by mohammed on 13/05/16.
 */
```

```
var utilUnit = require('./../model/utilUnit');
var addUnit = require('./../model/addUnit');


function removeRedundantMULOP(value){

    if(Array.isArray(value))
    {
        for(var i=0 ; i< value.length ; i++){
            if(value[i]=='0')
            { value.splice(i, 1);i--;}
            else break;
        }

    }
    else {

        for(var i=0 ; i< value.length ; i++){
            if(value.charAt(i)=='0')
            {value=utilUnit.replaceAt(value,i,'');i--}
            else break;
        }

    }
    return value;

}

function multBy_I_andExtendTo64Bit(operand,mutiplier,signed){

    switch (mutiplier){
        case '0' :
            if(operand.length == 32)
                return utilUnit.ZeroString32.concat(utilUnit.ZeroString32);
            else if (operand.length == 64)
                return utilUnit.ZeroString32.concat(utilUnit.ZeroString32);
            else  return utilUnit.ZeroString32.concat(utilUnit.ZeroString32)+'00';
            break ;

        case '1' :

            return  utilUnit.extendTo64Bit(operand,signed);
            break ;

        case '2' :

            return utilUnit.shiftLeft(utilUnit.extendTo64Bit(operand,signed),0);
            break ;

        case '-1' :


            return utilUnit.toTwoS_Complemnt(utilUnit.extendTo64Bit(operand,signed));
            break ;

        case '-2' :
            return
utilUnit.toTwoS_Complemnt(utilUnit.shiftLeft(utilUnit.extendTo64Bit(operand,signed),0)
);
            break ;
    }
}
```

```
function formattingMultiplicationOperands(operand1,operand2,Signed,NormalMul){
var OP1 =utilUnit.BitstringFormat(operand1);
    var OP2=utilUnit.BitstringFormat( operand2);
  return  '\n  '+(OP1.length <32&&!NormalMul
?'0':'')+OP1+utilUnit.toDecimal(operand1,Signed)+'* '+(OP2.length <32&&!NormalMul
?'0':'')+OP2+utilUnit.toDecimal(operand2,Signed)+utilUnit.shortLine.substring(5*utilUn
it.GET_REDUNDANT_BITS()/4,utilUnit.shortLine.length);
}


function normalMul32Bit(operand1,operand2,Signed){
    utilUnit.SET_REDUNDANT_BITS(
Math.min(utilUnit.CalculateRedendandBits(operand1),utilUnit.CalculateRedendandBits(ope
rand2)));
    var outputStirng=formattingMultiplicationOperands(operand1,operand2,Signed,true);


    utilUnit.SET_REDUNDANT_BITS(  64-(32-
Math.min(utilUnit.CalculateRedendandBits(operand1),utilUnit.CalculateRedendandBits(ope
rand2)))*2-2+(Signed?0:+2));

    operand2= removeRedundantMULOP(operand2); //reduction


    var partialProducts=new Array(operand2.length);


    for(var i = operand2.length-1 ; i >0 ; i--){

        partialProducts[i]=multBy_I_andExtendTo64Bit(operand1,
operand2.charAt(i),Signed);


        for (var j = 0;j<operand2.length-1-i ;j++){
            partialProducts[i]=  utilUnit.shiftLeft(partialProducts[i],0);
        }

    }

    //sign bit is treated differently according to signed systems with both negatives
    if ( Signed && operand2.length ==32 && operand2.charAt(0)=='1')
        partialProducts[0]=multBy_I_andExtendTo64Bit(operand1,'-1',Signed);
    else

partialProducts[0]=multBy_I_andExtendTo64Bit(operand1,operand2.charAt(0),Signed);


    for (var j = 0;j<operand2.length-1 ;j++){
        partialProducts[0]= utilUnit. shiftLeft(partialProducts[0],0);
    }

    return outputStirng+addUnit.multiAddition(partialProducts,Signed);
}



function BoothMul32Bit(operand1,operand2,Signed){

    utilUnit.SET_REDUNDANT_BITS(
Math.min(utilUnit.CalculateRedendandBits(operand1),utilUnit.CalculateRedendandBits(ope
rand2)));
```

```
    var OP1=utilUnit.BitstringFormat(operand1);
    var text1=    '\n  '+(OP1.length <32
?'0':'')+OP1+utilUnit.toDecimal(operand1,Signed);
    var outputString=formattingMultiplicationOperands(operand1,operand2,true);

    var RBM=utilUnit.GET_REDUNDANT_BITS();

    utilUnit.SET_REDUNDANT_BITS(  64-(32-
Math.min(utilUnit.CalculateRedendandBits(operand1),utilUnit.CalculateRedendandBits(ope
rand2)))*2);


    var partialProducts;
    if(!Signed)
     {
        operand1='0'+operand1;//n+1 system
        operand2='0'+operand2;//n+1 system

    }

    operand2 =BoothEndcoding(operand2);//generate the encoding
    operand2= removeRedundantMULOP(operand2); //reduction



    outputString +=text1+(' '+operand2.join(' ')).replace(/ 1/g, "
+1")+'(encoded)\n'+utilUnit.LINE.substring(RBM*4/5,utilUnit.LINE.length);

    partialProducts=[];

    for(var i = operand2.length-1 ; i >-1 ; i--){

        partialProducts[i]=multBy_I_andExtendTo64Bit(operand1, operand2[i],true);
        for (var j = 0;j<operand2.length-1-i ;j++){
            partialProducts[i]=  utilUnit.shiftLeft(partialProducts[i],0);
        }
    }

    return outputString+addUnit.multiAddition(partialProducts,true,!Signed);
}

function BitPairMul32Bit(operand1,operand2,Signed){
    utilUnit.SET_REDUNDANT_BITS(
Math.min(utilUnit.CalculateRedendandBits(operand1),utilUnit.CalculateRedendandBits(ope
rand2)));
    var OP1=utilUnit.BitstringFormat(operand1);
    var text1=    '\n  '+(OP1.length <32
?'0':'')+OP1+utilUnit.toDecimal(operand1,Signed);
    var outputStirng=formattingMultiplicationOperands(operand1,operand2,true);
    var RBM=utilUnit.GET_REDUNDANT_BITS();

    utilUnit.SET_REDUNDANT_BITS(  64-(32-
Math.min(utilUnit.CalculateRedendandBits(operand1),utilUnit.CalculateRedendandBits(ope
rand2)))*2);


    if(!Signed)
     {
        operand1='0'+operand1;//n+1 system
        operand2='0'+operand2;//n+1 system
    }
    operand2 =BitPairEndcoding(operand2);//generate the encoding
    operand2= removeRedundantMULOP(operand2); //reduction


    outputStirng +=text1+(' '+operand2.join(' ')).replace(/ 1/g, " +1").replace(/ 2/g,
" +2")+'(encoded)\n'+utilUnit.LINE.substring(RBM*4/5,utilUnit.LINE.length);


    var partialProducts=[];
    for(var i = operand2.length-1 ; i >-1 ; i--){
        partialProducts[i]=multBy_I_andExtendTo64Bit(operand1, operand2[i],true);

        for (var j = 0;j<operand2.length-1-i ;j++){
            partialProducts[i]= utilUnit. shiftLeft(partialProducts[i],0);
            partialProducts[i]= utilUnit. shiftLeft(partialProducts[i],0);
        }
    }

    return outputStirng+addUnit.multiAddition(partialProducts,true,!Signed );

}

function BoothEndcoding(operand){
    operand= operand+'0';
    var encoding =[];
    for(var i = operand.length-1 ; i >0 ; i--){
        switch (operand[i-1]+operand[i]){
            case '00':
            case '11':
                encoding[i-1]='0';
                break;
            case '01':
                encoding[i-1]='1';
                break;
            case '10':
                encoding[i-1]='-1';
                break;
        }
    }

    return encoding;
}


function BitPairEndcoding(operand){
    operand= operand+'0';

    if(operand.length % 2 == 0)
        operand=operand.charAt(0)+operand;

    var encoding =[];

    var j = Math.ceil(operand.length/2)-2;

    for(var i = operand.length-2;  i >0 ; i-=2,j--){
        switch (operand[i-1]+operand[i]+operand[i+1]){
            case '000':
            case '111':
                encoding[j]='0';
                break;
            case '001':
            case '010':
                encoding[j]='1';
                break;
            case '101':
            case '110':
                encoding[j]='-1';
                break;
            case '011':
                encoding[j]='2';
                break;
            case '100':
                encoding[j]='-2';
                break;
        }
    }

    return encoding;
}




module.exports = {
    normalMultplication:normalMul32Bit,
    multBy_I_andExtendTo64Bit:multBy_I_andExtendTo64Bit,
    BoothMultplication:BoothMul32Bit,
    BitPairMultplication:BitPairMul32Bit
};
```

divUnit.js

```
/**
 * Created by mohammed on 13/05/16.
 */

var utilUnit = require('./../model/utilUnit');
var addUnit = require('./../model/addUnit');
var operations={shift:'Shift\n',subtract:'Subtract\n',setQ:'Set
Qo\n',restore:'Restore\n',add:'Add\n'};

function RestoringDivision32Bit(dividend,divisor,Signed){
    var SignC1=false;
    var SignC2=false;
    //initialization
    utilUnit.SET_REDUNDANT_BITS(  utilUnit.CalculateRedendandBits(dividend) );
    var cycles=dividend.length-utilUnit.GET_REDUNDANT_BITS();
    var line=utilUnit.LINE.substring(0,cycles+3);

    if(Signed){

        if(dividend.charAt(0)=='1')//is negative
        {
            dividend=utilUnit.toTwoS_Complemnt(dividend);
            SignC1=true;
        }
        if(divisor.charAt(0)=='1')//is negative
        {
            divisor = utilUnit.toTwoS_Complemnt(divisor);
            SignC2=true;
        }

        if(SignC1 &&SignC2 )
        return RestoringDivision32Bit(dividend,divisor,false);
    }

var A,M,MTows;

    A='0'+utilUnit.ZeroString32;
    M='0'+divisor;
             MTows=utilUnit.toTwoS_Complemnt(M);

    var formattedOutput={left:'Initially \n
M\n',middle:utilUnit.BitstringFormat(A)+'\n'+utilUnit.BitstringFormat(M)+'\n',right:ut
ilUnit.BitstringFormat(dividend)+'\n\n'};


        for (var i=0 ; i <dividend.length ; i++){//each clock cycle

            A=utilUnit.shiftLeft(A,dividend.charAt(0));//shift step

            if( i > dividend.length-cycles-1){
            formattedOutput.left +=operations.shift;
            formattedOutput.middle +=utilUnit.BitstringFormat(A)+'\n';
            formattedOutput.right
+=utilUnit.BitstringFormat(dividend.substring(1,dividend.length))+'\n';
            }

            A= addUnit.addNBits (A,MTows,Signed,true/*do not update flags*/);
//subtract step

            if( i > dividend.length-cycles-1) {
                formattedOutput.left += operations.subtract + '\n';
                formattedOutput.middle += utilUnit.BitstringFormat(MTows) + '\n' +
line + '\n';
                formattedOutput.right += '\n\n';


                formattedOutput.left += operations.setQ;//set Qo
                formattedOutput.middle += utilUnit.BitstringFormat(A) + '\n';
                formattedOutput.right += '\n';
            }

            dividend=utilUnit.shiftLeft(dividend,utilUnit.invertBit(A.charAt(0))) ;


            if (A.charAt(0) =='1')//restore
            {
                A= addUnit.addNBits (A,M,Signed,true); //do not update flags


                if( i > dividend.length-cycles-1) {
                    formattedOutput.left += operations.restore + '\n';//restore
                    formattedOutput.middle += utilUnit.BitstringFormat(M) + '\n' +
line + '\n';
                    formattedOutput.right += '\n\n';

                }

            }

            if( i > dividend.length-cycles-1) {
                formattedOutput.left += '\n';
                formattedOutput.middle += utilUnit.BitstringFormat(A) + '\n';
                formattedOutput.right += utilUnit.BitstringFormat(dividend) + '\n';
            }
        }

    formattedOutput.left += '\t\n';
    formattedOutput.middle += 'Remainder ';  //Remainder at the last cycle
    formattedOutput.right += 'Quotient ' ; //Quotient at the last cycle
    var FinalOutput= {};

    FinalOutput.Text ='Restoring division is only concerned with unsigned integers.\n
';

    if((SignC1==true &&SignC2==false)||SignC1==false &&SignC2==true)
        FinalOutput.Text+='The actual Quotient
is\n'+utilUnit.BitstringFormat(utilUnit.toTwoS_Complemnt(dividend))+'\n';
    if(SignC1==true)
        FinalOutput.Text+='The actual Remainder
is\n'+utilUnit.BitstringFormat(utilUnit.toTwoS_Complemnt(A))+'\n';


    FinalOutput.Arith=formattedOutput;


return FinalOutput;

}


function NonRestoringDivision32Bit(dividend,divisor,Signed){

    //initialization

    var SignC1=false;
    var SignC2=false;

    utilUnit.SET_REDUNDANT_BITS(  utilUnit.CalculateRedendandBits(dividend) );
    var cycles=dividend.length-utilUnit.GET_REDUNDANT_BITS();
    var line=utilUnit.LINE.substring(0,cycles+3);



    if(Signed){
        if(dividend.charAt(0)=='1')//is negative
        {
            dividend=utilUnit.toTwoS_Complemnt(dividend);
            SignC1=true;
        }
        if(divisor.charAt(0)=='1')//is negative
        {
            divisor = utilUnit.toTwoS_Complemnt(divisor);
            SignC2=true;
        }

        if(SignC1 && SignC2 )
            return NonRestoringDivision32Bit(dividend,divisor,false);
    }


    var A,M,MTows;

    A='0'+utilUnit.ZeroString32;
    M='0'+divisor;
    MTows=utilUnit.toTwoS_Complemnt(M);
    var formattedOutput={left:'Initially \n
M\n',middle:utilUnit.BitstringFormat(A)+'\n'+utilUnit.BitstringFormat(M)+'\n',right:ut
ilUnit.BitstringFormat(dividend)+'\n\n'};


    for (var i=0 ; i <dividend.length ; i++) {//each clock cycle

        A = utilUnit.shiftLeft(A, dividend.charAt(0));//shift step

        if (i > dividend.length - cycles - 1) {


            formattedOutput.left += operations.shift;
            formattedOutput.right += utilUnit.BitstringFormat(dividend.substring(1,
dividend.length)) + '\n';
            if(i-dividend.length + cycles ==0 ){
                formattedOutput.middle += utilUnit.BitstringFormat(utilUnit.shiftLeft(
'0'+utilUnit.ZeroString32,dividend.charAt(0) ) ) + '\n';
            }else {
                formattedOutput.middle += utilUnit.BitstringFormat(A) + '\n';
            }


        }


        var tempAn =A.charAt(0) == '1'&&!(i-dividend.length + cycles ==0)  ;


        if (A.charAt(0) == '1')//add{
            A = addUnit.addNBits(A, M, Signed, true); //do not update flags
         else //sub
            A= addUnit.addNBits (A,MTows,Signed,true); //do not update flags

        dividend=utilUnit.shiftLeft(dividend,utilUnit.invertBit(A.charAt(0))) ;

        if( i > dividend.length-cycles-1) {
            formattedOutput.left += (tempAn?operations.add:operations.subtract) +
'\n';
            formattedOutput.middle += utilUnit.BitstringFormat( (tempAn?M:MTows)) +
'\n' + line + '\n';
            formattedOutput.right += '\n\n';


            formattedOutput.left += operations.setQ+'\n';//set Qo
            formattedOutput.middle += utilUnit.BitstringFormat(A) + '\n\n';
            formattedOutput.right += utilUnit.BitstringFormat(dividend) +'\n\n';
        }
    }


    if (A.charAt(0) =='1')//restore
        {
            A= addUnit.addNBits (A,M,Signed,true); //do not update flags

            formattedOutput.left += operations.restore + '\n';//restore
            formattedOutput.middle += utilUnit.BitstringFormat(M) + '\n' + line +
'\n';
            formattedOutput.right += '\n\n';

            formattedOutput.left += '\n';
            formattedOutput.middle += utilUnit.BitstringFormat(A) + '\n';
            formattedOutput.right += utilUnit.BitstringFormat(dividend) + '\n';
        }


    formattedOutput.left += '\t \n';
    formattedOutput.middle += 'Remainder ';  //Remainder at the last cycle
    formattedOutput.right += 'Quotient ' ; //Quotient at the last cycle

    var FinalOutput={};

    FinalOutput.Text ='Non-Restoring division is only concerned with unsigned
integers.\n ';

    if((SignC1==true &&SignC2==false)||SignC1==false &&SignC2==true)
        FinalOutput.Text+='The actual Quotient
is\n'+utilUnit.BitstringFormat(utilUnit.toTwoS_Complemnt(dividend))+'\n';
    if(SignC1==true)
        FinalOutput.Text+='The actual Remainder
is\n'+utilUnit.BitstringFormat(utilUnit.toTwoS_Complemnt(A))+'\n';


    FinalOutput.Arith=formattedOutput;


    return FinalOutput;
}


module.exports = {
    NonRestoringDivision32Bit:NonRestoringDivision32Bit,
    RestoringDivision32Bit:RestoringDivision32Bit
};
```

computations.js

```
/**
 * Created by mohammed on 27/04/16.
 */

var divUnit = require('./../model/divUnit');
var mulUnit = require('./../model/mulUnit');
var addUnit = require('./../model/addUnit');
var utilUnit = require('./../model/utilUnit');


module.exports = {
    Subtraction:addUnit.Subtraction,
    Addition:addUnit.Addition,
    toBitString32 :utilUnit.toBitString32,
    NormalMultplication:mulUnit.normalMultplication,
    BoothMultplication:mulUnit.BoothMultplication,
    BitPairMultplication:mulUnit.BitPairMultplication,

    RestoringDivision32Bit:divUnit.RestoringDivision32Bit,
    NonRestoringDivision32Bit:divUnit.NonRestoringDivision32Bit,
    flagsF :utilUnit.flagsF
};
```

frontEndStuff.js

```
/**
 * Created by mohammed on 25/04/16.
 *
 *
 * http://stackoverflow.com/questions/35783797/set-material-design-lite-radio-button-
option-with-jquery
 */


var RHSlist= document.getElementById("operations_list").getElementsByTagName("input");
var LHSlist= document.getElementById("LHS_parameters").getElementsByTagName("input");

var base =2;
var operNumber;

LHSlist[2].onclick= function() { base=2;  };
LHSlist[3].onclick= function() { base=10; };
LHSlist[4].onclick= function() { base=8;  };
LHSlist[5].onclick= function() { base=16; };



for(var i=0 ; i < RHSlist.length;i++){
    RHSlist[i].onclick= function() {operNumber =i;};
}


RHSlist[1].onclick= function() {operNumber =1;
LHSlist[6].parentNode.MaterialSwitch.on();};

LHSlist[6].onclick= function() { if(operNumber==1)
LHSlist[6].parentNode.MaterialSwitch.on(); };


function isBinary(field){
    for(var i=0 ; i<field.length ; i++ ){
        var charCode=field.charCodeAt(i);
        if (  !(charCode ==48 || charCode ==49 ))
           return false;
    }
    return true;
}

function isDecimal(field){
    for(var i=0 ; i<field.length ; i++ ){
        var charCode=field.charCodeAt(i);
        if (  !(charCode > 47 && charCode < 58 ))
            return false;
    }
    return true;
}
```

```
function isOctal(field){
    for(var i=0 ; i<field.length ; i++ ){
        var charCode=field.charCodeAt(i);
        if (  !(charCode > 47 && charCode < 56 ))
            return false;
    }
    return true;
}
function isOverflow(Signed,field){


    switch (base){
        case 8:
            if(field.length > 10)
                return true;
            break;
        case 10:
            if(!((field < (Math.pow(2,32)-1)&& !Signed)  ||(    field <
(Math.pow(2,31)-1) && field > -(Math.pow(2,31)) && Signed ) ))
                return true;
            break;
        case 16:
            if(field.length > 8 )
            return true;
            break;
    }

    return false;
}

function isValidNumbers(){

    var field1=LHSlist[0].value;
    var field2=LHSlist[1].value;

    if(LHSlist[6].checked){
       if(field1.charAt(0)=='-') field1= field1.substring(1, field1.length) ;
       if(field2.charAt(0)=='-') field2= field2.substring(1, field2.length) ;
    }



    if(!(field1.length && field2.length) )
        return false;


    var IsValid1=true;
    var IsValid2=true;
    switch (base){
        case 2:
            IsValid1=isBinary(field1);
            IsValid2=isBinary(field2);
            break;
        case 8:
            IsValid1=isOctal(field1);
            IsValid2=isOctal(field2);
            break;
        case 10:
            IsValid1=isDecimal(field1);
            IsValid2=isDecimal(field2);
            break;
    }
    return IsValid1 && IsValid2;
}


function isValidKey(evt,trigger){

    var charCode = (evt.which) ? evt.which : event.keyCode;

    field =  Number(trigger.value+String.fromCharCode(charCode));


    if(  LHSlist[6].checked && charCode==45 && trigger.value.charAt(0) !='-'
        &&
        ( (trigger.value.length < 32 && base==2)||
            (base==16 && trigger.value.length < 8) ||
            (base==8 && trigger.value.length < 10) ||
            (base==10 && (((field < (Math.pow(2,32)-1)&& !LHSlist[6].checked)  && (
field < (Math.pow(2,31)-1) && field > -(Math.pow(2,31)) &&  LHSlist[6].checked  ) )) )
        )

    )

    {  trigger.value='-'+trigger.value; return false;}


    else if(base==2 && trigger.value.length > 31)
        return false;
    else if(base==16 && trigger.value.length > 7)
        return false;
    else if(base==8 && trigger.value.length > 9)
        return false;
    else if(base==10 && (!((field < (Math.pow(2,32)-1)&& !LHSlist[6].checked)  ||(
field < (Math.pow(2,31)-1) && field > -(Math.pow(2,31))&&  LHSlist[6].checked    ) ))
)
        return false;

    if (  (charCode ==48 || charCode ==49) && base==2  )
        return true;

    else  if (charCode > 47 && charCode < 58 && base==10 )
        return true;
    else if  (charCode > 47 && charCode < 56 && base==8 )
        return true;

    else if  ( ((charCode > 64 && charCode < 71 )  || (charCode > 47 && charCode < 58)
)  && base==16 )
        return true;

    return false;
}

function Getparameters(){

    for (operNumber=0 ;operNumber <7;operNumber++)
        if(RHSlist[operNumber].checked)
            break;
    return {Operation:operNumber,
        Field1 : LHSlist[0].value,
        Field2 : LHSlist[1].value,
        Base:base,
        Signed:LHSlist[6].checked
    }
```

```
}

function ComputationRequest(){
    var params=Getparameters();

    if(isOverflow(params.Signed,params.Field1) ||
isOverflow(params.Signed,params.Field2)  ){
       return;
    }

    if(isValidNumbers()){
    $.post("/", //jquery
        { params:JSON.stringify(params)},
        function(data, status){//callback
            if(status==='success')
            {
                    $('#output').empty();

                if(params.Operation>4)//division
                {
                    $("#output").append('<table > <tr id="outputBars">
</tr></table>').css('margin-left',
((data.Arith.middle.substring(1,data.Arith.middle.length).indexOf('\n'))*-
.95+40)+''+'%');

                    $("#outputBars").append($("<td id='operDiv'></td>") .html( '<pre
width="30">' + data.Arith.left.replace(/\n/g, "<br />"  )+ '</pre>' ));
                    $("#outputBars").append($("<td></td>") .html('<pre width="30">' +
data.Arith.middle.replace(/\n/g, "<br />")+ '</pre>'));
                    $("#outputBars").append($("<td></td>") .html( '<pre width="30">' +
data.Arith.right.replace(/\n/g, "<br />")+ '</pre>'));


                    $("pre") .hover(function(){
                        $(this).css("background-color", "#ddd");
                    }, function(){
                        $(this).css("background-color", "white");
                    });

                }else//not division
                {

                    $("#output").html('<pre width="30">' + data.Arith.replace(/\n/g,
"<br />") + '</pre>').css('margin-left',
(((data.Arith.substring(1,data.Arith.length).indexOf('\n'))))*-.87+48.7+''+'%');
                    $("pre") .hover(null,null);
                }
               $('p').html(data.Text.replace(/\t/g,
'&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;'));
            }
        });
    }else {
        var snackbarContainer = document.querySelector('#demo-toast-example');
            var data = {message: 'Check your inputs'};
            snackbarContainer.MaterialSnackbar.showSnackbar(data);
    }
}

function Reset(){
    LHSlist[0].value=null;
    LHSlist[1].value=null;
    $("#output").empty();
    $('p').empty();
    RHSlist[0].parentNode.MaterialRadio.check();
    for (i=1 ;i <7;i++)
        RHSlist[i].parentNode.MaterialRadio.uncheck();
    LHSlist[2].parentNode.MaterialRadio.check();
    LHSlist[3].parentNode.MaterialRadio.uncheck();
    LHSlist[4].parentNode.MaterialRadio.uncheck();
    LHSlist[5].parentNode.MaterialRadio.uncheck();
    LHSlist[6].parentNode.MaterialSwitch.on();
}


function  Dialog(){
    var snackbarContainer = document.querySelector('#demo-toast-example');
    var data = {message: 'This web application is an interactive tool, which simulates 
Boolean arithmetics with detailed steps .    Credits: Mohammed Alaa el komy   '
,timeout: 7000};
    snackbarContainer.MaterialSnackbar.showSnackbar(data);
}
```

Frontend

styles.css

```
td{
  padding-top: 3%;
  padding-bottom: 3%;
}

pre{
  padding: -3%;
}

.demo-ribbon {
  width: 100%;
  height: 40vh;
  background-color: #673AB7;
  -webkit-flex-shrink: 0;
      -ms-flex-negative: 0;
          flex-shrink: 0;
}

.demo-main {
  margin-top: -35vh;
  -webkit-flex-shrink: 0;
      -ms-flex-negative: 0;
          flex-shrink: 0;
}

.demo-header .mdl-layout__header-row {
  padding-left: 40px;
}

.demo-container {
  max-width: 1500px;
  width: calc(100% - 16px);
  margin: 0 auto;
}

.demo-content {

  border-radius: 2px;
  padding: 60px 10px;
  margin-bottom: 80px;
}

.demo-layout.is-small-screen .demo-content {
  padding: 40px 28px;
}



.demo-footer {
  padding-left: 40px;
}

.demo-footer .mdl-mini-footer--link-list a {
  font-size: 13px;
}

#view-source {
  position: fixed;
  display: block;
  right: 0;
  bottom: 0;
  margin-right: 40px;
  margin-bottom: 40px;
  z-index: 900;
}


.mdl-radio
{
  display: inline;
}
.operation-list{
  margin-right: 10px;
  width: 270px;

}

.input-items
{
  width: 270px;
  margin-left: 20px;
}
```

```
.mdl-button
{
  width: 100px;
  margin-left: 10px;
}
```

home.ejs

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-
scale=1.0">
    <title>Logic simulator</title>

    <link rel="stylesheet" href="./stylesheets/material.fonts.css">
    <link rel="stylesheet" type="text/css" href="./stylesheets/material.min.css">
    <link rel="stylesheet" href="./stylesheets/styles.css">

</head>
<body>

<div id="demo-toast-example" class="mdl-js-snackbar mdl-snackbar">
    <div class="mdl-snackbar__text"></div>
    <button class="mdl-snackbar__action" type="button"></button>
</div>




<div class="demo-layout mdl-layout mdl-layout--fixed-header mdl-js-layout mdl-color--
grey-100">
    <header class="demo-header mdl-layout__header mdl-layout__header--scroll mdl-
color--grey-100 mdl-color-text--grey-800">
        <div class="mdl-layout__header-row">
            <span class="mdl-layout-title">Logic simulator</span><div class="mdl-
layout-spacer"></div>

            <!--white holder -->
            <div class="mdl-textfield mdl-js-textfield mdl-textfield--expandable">
                <div class="mdl-textfield__expandable-holder">
                    <!--white holder -->
                </div>
            </div>
        </div>
    </header>

    <div class="demo-ribbon"></div>
    <main class="demo-main mdl-layout__content">
        <div class="demo-container mdl-grid">
            <div class="mdl-cell mdl-cell--2-col mdl-cell--hide-tablet mdl-cell--hide-
phone"></div>
            <div class="demo-content mdl-color--white mdl-shadow--4dp content mdl-
color-text--grey-800 mdl-cell mdl-cell--8-col">

                <form action="#">






                    <ul id="operations_list" class="mdl-list operation-list"
style="margin-left:50px;float:right;float:top;">

                        <li class="mdl-list__item">
                            <span class="mdl-list__item-primary-
content">Addition</span>
                                <span class="mdl-list__item-secondary-action">
                                     <label id="mamam" class="mdl-radio  mdl-js-radio
mdl-js-ripple-effect" for="operation-1">
                                         <input type="radio" id="operation-1"
class="mdl-radio__button" name="options2"  checked />
                                     </label>
                        </span>
                        </li>

                        <li class="mdl-list__item">
                            <span class="mdl-list__item-primary-
content">Subtraction</span>
                                <span class="mdl-list__item-secondary-action">
                                     <label class="mdl-radio  mdl-js-radio mdl-js-
ripple-effect" for="operation-2">
                                         <input type="radio" id="operation-2"
class="mdl-radio__button" name="options2"  />
                                     </label>
                        </span>
                        </li>

                        <li class="mdl-list__item">
                            <span class="mdl-list__item-primary-content">Normal
Mult</span>
                                <span class="mdl-list__item-secondary-action">
                                     <label class="mdl-radio  mdl-js-radio mdl-js-
ripple-effect" for="operation-3">
                                         <input type="radio" id="operation-3"
class="mdl-radio__button" name="options2"   />
                                     </label>
                        </span>
                        </li>

                        <li class="mdl-list__item">
                            <span class="mdl-list__item-primary-content">Booth
```

```
Mult</span>
                                <span class="mdl-list__item-secondary-action">
                                     <label class="mdl-radio  mdl-js-radio mdl-js-
ripple-effect" for="operation-4">
                                         <input type="radio" id="operation-4"
class="mdl-radio__button" name="options2"   />
                                     </label>
                        </span>
                        </li>

                        <li class="mdl-list__item">
                            <span class="mdl-list__item-primary-content">Bit-pair
Mult</span>
                                <span class="mdl-list__item-secondary-action">
                                     <label class="mdl-radio  mdl-js-radio mdl-js-
ripple-effect" for="operation-5">
                                         <input type="radio" id="operation-5"
class="mdl-radio__button" name="options2"   />
                                     </label>
                        </span>
                        </li>

                        <li class="mdl-list__item">
                            <span class="mdl-list__item-primary-content">Restoring
division</span>
                                <span class="mdl-list__item-secondary-action">
                                     <label class="mdl-radio  mdl-js-radio mdl-js-
ripple-effect" for="operation-6">
                                         <input type="radio" id="operation-6"
class="mdl-radio__button" name="options2"   />
                                     </label>
                        </span>
                        </li>

                        <li class="mdl-list__item">
                            <span class="mdl-list__item-primary-content">Non-restoring
Division</span>
                                <span class="mdl-list__item-secondary-action">
                                     <label class="mdl-radio  mdl-js-radio mdl-js-
ripple-effect" for="operation-7">
                                         <input type="radio" id="operation-7"
class="mdl-radio__button" name="options2"  />
                                     </label>
                        </span>
                        </li>


                    </ul>




<span>
                <ul class="mdl-list input-items" id="LHS_parameters">


                    <!-- text entries -->
                    <li class="mdl-list__item"  style=" width:600px;" >
                        <div class="mdl-textfield mdl-js-textfield mdl-textfield--
floating-label">
                            <input  class="mdl-textfield__input" type="n"
id="operand1" onkeypress="return isValidKey(event,this)">
                            <label class="mdl-textfield__label" for="operand1">First
```

```
Operand</label>
                        </div>
                    </li>

                    <li class="mdl-list__item" style=" width:600px;" >
                        <div class="mdl-textfield mdl-js-textfield mdl-textfield--
floating-label">
                            <input  class="mdl-textfield__input" type="n"
id="operand2" onkeypress="return isValidKey(event,this)">
                            <label class="mdl-textfield__label" for="operand2">Second
Operand</label>

                        </div>
                    </li>


                    <li class="mdl-list__item"  >
                        <label class="mdl-list__item-primary-content mdl-radio mdl-js-
radio mdl-js-ripple-effect" for="option-1">
                            <input type="radio" id="option-1" class="mdl-
radio__button" name="options"  checked>
                            <span class="mdl-radio__label">Binary</span>
                        </label >


                        <label class="mdl-list__item-secondary-action  mdl-radio mdl-
js-radio mdl-js-ripple-effect" for="option-2" >
                            <input type="radio" id="option-2" class="mdl-
radio__button" name="options" >
                            <span class="mdl-radio__label" style="padding-right:
1px">Decimal
                                &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</span>
                        </label >
                    </li>

                    <li class="mdl-list__item"  >
                        <label class="mdl-list__item-primary-content mdl-radio mdl-js-
radio mdl-js-ripple-effect" for="option-3">
                            <input type="radio" id="option-3" class="mdl-
radio__button" name="options" >
                            <span class="mdl-radio__label">Octal</span>
                        </label >


                        <label class="mdl-list__item-secondary-action  mdl-radio mdl-
js-radio mdl-js-ripple-effect" for="option-4">
                            <input type="radio" id="option-4" class="mdl-
radio__button" name="options" >
                            <span class="mdl-radio__label">Hexadecimal</span>
                        </label >
                    </li>

                    <li class="mdl-list__item" style=" width:170px; margin-left: 50px;
margin-top: 15px;    " >



                        <span >
                            <label class="mdl-switch mdl-js-switch mdl-js-ripple-
effect" for="list-switch-1">
                                <input type="checkbox" id="list-switch-1" class="mdl-
switch__input" checked />
                            </label>
                        </span>

                        <span class="mdl-list__item-secondary-content" style="margin-
left: 45px;">Signed</span>


                    </li>



                    <li class="mdl-list__item" style="margin-top: 15px;">

                        <button  type="button" class=" mdl-button mdl-js-button mdl-
button--raised mdl-js-ripple-effect mdl-button--colored"
onclick="ComputationRequest()">
                            Compute
                        </button>

                        <button type="button" class=" mdl-button mdl-js-button mdl-
button--raised mdl-js-ripple-effect mdl-button--colored" onclick="Reset()">
                            Reset
                        </button>
                    </li>
                </ul>
</span>


                </form>


            <div id="output" style="font-size: 18px" >    <!-- output text --></div>


                <p id="outputText" style='text-align: center;font-size: 16px'></p>


            </div>


        </div>




        <footer class="demo-footer mdl-mini-footer">
            <div class="mdl-mini-footer--left-section">
                <ul class="mdl-mini-footer--link-list">
                    <li><a target="_blank" href=
"http://localhost:3000/f/help.png">Help</a></li>
                    <li><a target="_blank" href=
"http://localhost:3000/f/tech.pdf">Technical implementation </a></li>
                    <li><a href= "javascript:;" onclick="Dialog()">About this
project</a></li>
                </ul>
            </div>
        </footer>



    </main>

</div>
```

```
<a href="http://localhost:3000/f/source.pdf" target="_blank" id="view-source"
class="mdl-button mdl-js-button mdl-button--raised mdl-js-ripple-effect mdl-color--
accent mdl-color-text--accent-contrast">View Source</a>

<script src="./javascripts/material.min.js"></script>
<script src="./javascripts/frontEndStuff.js"></script>
<script src="./javascripts/jquery.min.js"></script>

</body>
</html>
```

error.ejs

```
<h1><%= message %></h1>
<h2><%= error.status %></h2>
<pre><%= error.stack %></pre>
```