The bigintcalc package

Heiko Oberdiek*

2019/12/15 v1.5

Abstract

This package provides expandable arithmetic operations with big integers that can exceed TEX's number limits.

Contents

*Please report any issues at https://github.com/ho-tex/bigintcalc/issues

1 Documentation

1.1 Introduction

Package bigintcalc defines arithmetic operations that deal with big integers. Big integers can be given either as explicit integer number or as macro code that expands to an explicit number. *Big* means that there is no limit on the size of the number. Big integers may exceed TEX's range limitation of -2147483647 and 2147483647. Only memory issues will limit the usable range.

In opposite to package intcalc unexpandable command tokens are not supported, even if they are valid TEX numbers like count registers or commands created by `\chardef`. Nevertheless they may be used, if they are prefixed by `\number`.

Also ε-TEX's `\numexpr` expressions are not supported directly in the manner of package intcalc. However they can be given if `\the\numexpr` or `\number\numexpr` are used.

The operations have the form of macros that take one or two integers as parameter and return the integer result. The macro name is a three letter operation name prefixed by the package name, e.g. `\bigintcalcAdd{10}{43}` returns 53.

The macros are fully expandable, exactly two expansion steps generate the result. Therefore the operations may be used nearly everywhere in TEX, even inside `\csname`, file names, or other expandable contexts.

1.2 Conditions

1.2.1 Preconditions

- Arguments can be anything that expands to a number that consists of optional signs and digits.

- The arguments and return values must be sound. Zero as divisor or factorials of negative numbers will cause errors.

1.2.2 Postconditions

Additional properties of the macros apart from calculating a correct result (of course ☺):

- The macros are fully expandable. Thus they can be used inside `\edef`, `\csname`, for example.
- Furthermore exactly two expansion steps calculate the result.
- The number consists of one optional minus sign and one or more digits. The first digit is larger than zero for numbers that consists of more than one digit.

  In short, the number format is exactly the same as `\number` generates, but without its range limitation. And the tokens (minus sign, digits) have catcode 12 (other).
- Call by value is simulated. First the arguments are converted to numbers. Then these numbers are used in the calculations.

  Remember that arguments may contain expensive macros or ε-TEX expressions. This strategy avoids multiple evaluations of such arguments.

1.3 Error handling

Some errors are detected by the macros, example: division by zero. In this cases an undefined control sequence is called and causes a TeX error message, example: `\BigIntCalcError:DivisionByZero`. The name of the control sequence contains the reason for the error. The TEX error may be ignored. Then the operation returns zero as result. Because the macros are supposed to work in expandible contexts. An traditional error message, however, is not expandable and would break these contexts.

1.4 Operations

Some definition equations below use the function Int that converts a real number to an integer. The number is truncated that means rounding to zero:

$$\mathrm{Int}(x) := \begin{cases} \lfloor x \rfloor & \text{if } x \geq 0 \\ \lceil x \rceil & \text{otherwise} \end{cases}$$

1.4.1 Num

`\bigintcalcNum` {⟨*x*⟩}

Macro `\bigintcalcNum` converts its argument to a normalized integer number without unnecessary leading zeros or signs. The result matches the regular expression:

```
0|-?[1-9][0-9]*
```

1.4.2 Inv, Abs, Sgn

`\bigintcalcInv {⟨x⟩}`

Macro `\bigintcalcInv` switches the sign.

$$\mathrm{Inv}(x) := -x$$

`\bigintcalcAbs {⟨x⟩}`

Macro `\bigintcalcAbs` returns the absolute value of integer $\langle x \rangle$.

$$\mathrm{Abs}(x) := |x|$$

`\bigintcalcSgn {⟨x⟩}`

Macro `\bigintcalcSgn` encodes the sign of $\langle x \rangle$ as number.

$$\mathrm{Sgn}(x) := \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

These return values can easily be distinguished by `\ifcase`:

```
\ifcase\bigintcalcSgn{<x>}
  $x=0$
\or
  $x>0$
\else
  $x<0$
\fi
```

1.4.3 Min, Max, Cmp

`\bigintcalcMin {⟨x⟩} {⟨y⟩}`

Macro `\bigintcalcMin` returns the smaller of the two integers.

$$\mathrm{Min}(x, y) := \begin{cases} x & \text{if } x < y \\ y & \text{otherwise} \end{cases}$$

`\bigintcalcMax {⟨x⟩} {⟨y⟩}`

Macro `\bigintcalcMax` returns the larger of the two integers.

$$\mathrm{Max}(x, y) := \begin{cases} x & \text{if } x > y \\ y & \text{otherwise} \end{cases}$$

`\bigintcalcCmp` {⟨*x*⟩} {⟨*y*⟩}

Macro `\bigintcalcCmp` encodes the comparison result as number:

$$\mathrm{Cmp}(x,y) := \begin{cases} -1 & \text{if } x < y \\ 0 & \text{if } x = y \\ 1 & \text{if } x > y \end{cases}$$

These values can be distinguished by `\ifcase`:

```
\ifcase\bigintcalcCmp{<x>}{<y>}
  $x=y$
\or
  $x>y$
\else
  $x<y$
\fi
```

1.4.4 Odd

`\bigintcalcOdd` {⟨*x*⟩}

$$\mathrm{Odd}(x) := \begin{cases} 1 & \text{if } x \text{ is odd} \\ 0 & \text{if } x \text{ is even} \end{cases}$$

1.4.5 Inc, Dec, Add, Sub

`\bigintcalcInc` {⟨*x*⟩}

Macro `\bigintcalcInc` increments ⟨*x*⟩ by one.

$$\mathrm{Inc}(x) := x + 1$$

`\bigintcalcDec` {⟨*x*⟩}

Macro `\bigintcalcDec` decrements ⟨*x*⟩ by one.

$$\mathrm{Dec}(x) := x - 1$$

`\bigintcalcAdd` {⟨*x*⟩} {⟨*y*⟩}

Macro `\bigintcalcAdd` adds the two numbers.

$$\mathrm{Add}(x,y) := x + y$$

`\bigintcalcSub` {⟨*x*⟩} {⟨*y*⟩}

Macro `\bigintcalcSub` calculates the difference.

$$\mathrm{Sub}(x,y) := x - y$$

1.4.6 Shl, Shr

`\bigintcalcShl` {⟨x⟩}

Macro `\bigintcalcShl` implements shifting to the left that means the number is multiplied by two. The sign is preserved.

$$\mathrm{Shl}(x) := x * 2$$

`\bigintcalcShr` {⟨x⟩}

Macro `\bigintcalcShr` implements shifting to the right. That is equivalent to an integer division by two. The sign is preserved.

$$\mathrm{Shr}(x) := \mathrm{Int}(x/2)$$

1.4.7 Mul, Sqr, Fac, Pow

`\bigintcalcMul` {⟨x⟩} {⟨y⟩}

Macro `\bigintcalcMul` calculates the product of ⟨x⟩ and ⟨y⟩.

$$\mathrm{Mul}(x, y) := x * y$$

`\bigintcalcSqr` {⟨x⟩}

Macro `\bigintcalcSqr` returns the square product.

$$\mathrm{Sqr}(x) := x^2$$

`\bigintcalcFac` {⟨x⟩}

Macro `\bigintcalcFac` returns the factorial of ⟨x⟩. Negative numbers are not permitted.

$$\mathrm{Fac}(x) := x! \qquad \text{for } x \geq 0$$

$(0! = 1)$

`\bigintcalcPow` Mx My

Macro `\bigintcalcPow` calculates the value of ⟨x⟩ to the power of ⟨y⟩. The error "division by zero" is thrown if ⟨x⟩ is zero and ⟨y⟩ is negative. permitted:

$$\mathrm{Pow}(x, y) := \mathrm{Int}(x^y) \qquad \text{for } x \neq 0 \text{ or } y \geq 0$$

$(0^0 = 1)$

1.4.8 Div, Mul

`\bigintcalcDiv` {⟨*x*⟩} {⟨*y*⟩}

Macro `\bigintcalcDiv` performs an integer division. Argument ⟨*y*⟩ must not be zero.

$$\mathrm{Div}(x, y) := \mathrm{Int}(x/y) \qquad \text{for } y \neq 0$$

`\bigintcalcMod` {⟨*x*⟩} {⟨*y*⟩}

Macro `\bigintcalcMod` gets the remainder of the integer division. The sign follows the divisor ⟨*y*⟩. Argument ⟨*y*⟩ must not be zero.

$$\mathrm{Mod}(x, y) := x \,\%\, y \qquad \text{for } y \neq 0$$

The result ranges:

$$-|y| < \mathrm{Mod}(x, y) \leq 0 \qquad \text{for } y < 0$$
$$0 \leq \mathrm{Mod}(x, y) < y \qquad \text{for } y \geq 0$$

1.5 Interface for programmers

If the programmer can ensure some more properties about the arguments of the operations, then the following macros are a little more efficient.

In general numbers must obey the following constraints:

- Plain number: digit tokens only, no command tokens.
- Non-negative. Signs are forbidden.
- Delimited by exclamation mark. Curly braces around the number are not allowed and will break the code.

`\BigIntCalcOdd` ⟨*number*⟩ `!`

`1/0` is returned if ⟨*number*⟩ is odd/even.

`\BigIntCalcInc` ⟨*number*⟩ `!`

Incrementation.

`\BigIntCalcDec` ⟨*number*⟩ `!`

Decrementation, positive number without zero.

`\BigIntCalcAdd` ⟨*number A*⟩ `!` ⟨*number B*⟩ `!`

Addition, $A \geq B$.

`\BigIntCalcSub` ⟨*number A*⟩ `!` ⟨*number B*⟩ `!`

Subtraction, $A \geq B$.

`\BigIntCalcShl` ⟨*number*⟩ `!`

Left shift (multiplication with two).

`\BigIntCalcShr` ⟨*number*⟩ `!`

Right shift (integer division by two).

`\BigIntCalcMul` ⟨*number A*⟩ `!` ⟨*number B*⟩ `!`

Multiplication, $A \geq B$.

`\BigIntCalcDiv` ⟨*number A*⟩ `!` ⟨*number B*⟩ `!`

Division operation.

`\BigIntCalcMod` ⟨*number A*⟩ `!` ⟨*number B*⟩ `!`

Modulo operation.

2 Implementation

```
⟨*package⟩
```

2.1 Reload check and package identification

Reload check, especially if the package is not used with LaTeX.

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode35=6 % #
  \catcode39=12 % '
  \catcode44=12 % ,
  \catcode45=12 % -
  \catcode46=12 % .
  \catcode58=12 % :
  \catcode64=11 % @
  \catcode123=1 % {
  \catcode125=2 % }
  \expandafter\let\expandafter\x\csname ver@bigintcalc.sty\endcsname
  \ifx\x\relax % plain-TeX, first loading
  \else
    \def\empty{}%
    \ifx\x\empty % LaTeX, first loading,
      % variable is initialized, but \ProvidesPackage not yet seen
    \else
      \expandafter\ifx\csname PackageInfo\endcsname\relax
        \def\x#1#2{%
          \immediate\write-1{Package #1 Info: #2.}%
        }%
      \else
        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
      \fi
      \x{bigintcalc}{The package is already loaded}%
```

```
      \aftergroup\endinput
    \fi
  \fi
\endgroup%
```

Package identification:

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode35=6 % #
  \catcode39=12 % '
  \catcode40=12 % (
  \catcode41=12 % )
  \catcode44=12 % ,
  \catcode45=12 % -
  \catcode46=12 % .
  \catcode47=12 % /
  \catcode58=12 % :
  \catcode64=11 % @
  \catcode91=12 % [
  \catcode93=12 % ]
  \catcode123=1 % {
  \catcode125=2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@bigintcalc.sty\endcsname
\ProvidesPackage{bigintcalc}%
  [2019/12/15 v1.5 Expandable calculations on big integers (HO)]%
```

2.2 Catcodes

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode123=1 % {
  \catcode125=2 % }
  \catcode64=11 % @
  \def\x{\endgroup
    \expandafter\edef\csname BIC@AtEnd\endcsname{%
      \endlinechar=\the\endlinechar\relax
      \catcode13=\the\catcode13\relax
      \catcode32=\the\catcode32\relax
      \catcode35=\the\catcode35\relax
      \catcode61=\the\catcode61\relax
      \catcode64=\the\catcode64\relax
```

```
      \catcode123=\the\catcode123\relax
      \catcode125=\the\catcode125\relax
    }%
  }%
\x\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\BIC@AtEnd{%
    \BIC@AtEnd
    \catcode#1=\the\catcode#1\relax
  }%
  \catcode#1=#2\relax
}
\TMP@EnsureCode{33}{12}% !
\TMP@EnsureCode{36}{14}% $ (comment!)
\TMP@EnsureCode{38}{14}% & (comment!)
\TMP@EnsureCode{40}{12}% (
\TMP@EnsureCode{41}{12}% )
\TMP@EnsureCode{42}{12}% *
\TMP@EnsureCode{43}{12}% +
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{58}{11}% : (letter!)
\TMP@EnsureCode{60}{12}% <
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{63}{14}% ? (comment!)
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{93}{12}% ]
\edef\BIC@AtEnd{\BIC@AtEnd\noexpand\endinput}
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname BIC@TestMode\endcsname\relax
\else
  \catcode63=9 % ? (ignore)
\fi
? \let\BIC@@TestMode\BIC@TestMode
```

2.3 ε-TEX detection

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname numexpr\endcsname\relax
  \catcode36=9 % $ (ignore)
\else
  \catcode38=9 % & (ignore)
\fi
```

2.4 Help macros

\BIC@Fi

```
\let\BIC@Fi\fi
```

\BIC@AfterFi

```
\def\BIC@AfterFi#1#2\BIC@Fi{\fi#1}%
```

```
\BIC@AfterFiFi
\def\BIC@AfterFiFi#1#2\BIC@Fi{\fi\fi#1}%
\BIC@AfterFiFiFi
\def\BIC@AfterFiFiFi#1#2\BIC@Fi{\fi\fi\fi#1}%
\BIC@Space
\begingroup
  \def\x#1{\endgroup
    \let\BIC@Space= #1%
  }%
\x{ }
```

2.5 Expand number

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \def\TMP@RequirePackage#1[#2]{%
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname ver@#1.sty\endcsname\relax
      \input #1.sty\relax
    \fi
  }%
  \TMP@RequirePackage{pdftexcmds}[2007/11/11]%
\else
  \RequirePackage{pdftexcmds}[2007/11/11]%
\fi
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname pdf@escapehex\endcsname\relax
\BIC@Expand
  \def\BIC@Expand#1{%
    \romannumeral0%
    \BIC@@Expand#1!\@nil{}%
  }%
\BIC@@Expand
  \def\BIC@@Expand#1#2\@nil#3{%
    \expandafter\ifcat\noexpand#1\relax
      \expandafter\@firstoftwo
    \else
      \expandafter\@secondoftwo
    \fi
    {%
      \expandafter\BIC@@Expand#1#2\@nil{#3}%
    }{%
      \ifx#1!%
        \expandafter\@firstoftwo
      \else
        \expandafter\@secondoftwo
      \fi
      { #3}{%
        \BIC@@Expand#2\@nil{#3#1}%
      }%
    }%
  }%
```

\@firstoftwo

```
  \expandafter\ifx\csname @firstoftwo\endcsname\relax
    \long\def\@firstoftwo#1#2{#1}%
  \fi
```

\@secondoftwo

```
  \expandafter\ifx\csname @secondoftwo\endcsname\relax
    \long\def\@secondoftwo#1#2{#2}%
  \fi
\else
```

\BIC@Expand

```
  \def\BIC@Expand#1{%
    \romannumeral0\expandafter\expandafter\expandafter\BIC@Space
    \pdf@unescapehex{%
      \expandafter\expandafter\expandafter
      \BIC@StripHexSpace\pdf@escapehex{#1}20\@nil
    }%
  }%
```

\BIC@StripHexSpace

```
  \def\BIC@StripHexSpace#120#2\@nil{%
    #1%
    \ifx\\#2\\%
    \else
      \BIC@AfterFi{%
        \BIC@StripHexSpace#2\@nil
      }%
    \BIC@Fi
  }%
\fi
```

2.6 Normalize expanded number

\BIC@Normalize

#1: result sign
#2: first token of number

```
\def\BIC@Normalize#1#2{%
  \ifx#2-%
    \ifx\\#1\\%
      \BIC@AfterFiFi{%
        \BIC@Normalize-%
      }%
    \else
      \BIC@AfterFiFi{%
        \BIC@Normalize{}%
      }%
    \fi
  \else
    \ifx#2+%
      \BIC@AfterFiFi{%
        \BIC@Normalize{#1}%
      }%
    \else
      \ifx#20%
        \BIC@AfterFiFiFi{%
          \BIC@NormalizeZero{#1}%
```

```
        }%
      \else
        \BIC@AfterFiFiFi{%
          \BIC@NormalizeDigits#1#2%
        }%
      \fi
    \fi
  \BIC@Fi
}
```

\BIC@NormalizeZero

```
\def\BIC@NormalizeZero#1#2{%
  \ifx#2!%
    \BIC@AfterFi{ 0}%
  \else
    \ifx#20%
      \BIC@AfterFiFi{%
        \BIC@NormalizeZero{#1}%
      }%
    \else
      \BIC@AfterFiFi{%
        \BIC@NormalizeDigits#1#2%
      }%
    \fi
  \BIC@Fi
}
```

\BIC@NormalizeDigits

```
\def\BIC@NormalizeDigits#1!{ #1}
```

2.7 Num

\bigintcalcNum

```
\def\bigintcalcNum#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Normalize
  \expandafter\expandafter\expandafter{%
  \expandafter\expandafter\expandafter}%
  \BIC@Expand{#1}!%
}
```

2.8 Inv, Abs, Sgn

\bigintcalcInv

```
\def\bigintcalcInv#1{%
  \romannumeral0\expandafter\expandafter\expandafter\BIC@Space
  \bigintcalcNum{-#1}%
}
```

\bigintcalcAbs

```
\def\bigintcalcAbs#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Abs
  \bigintcalcNum{#1}%
}
```

\BIC@Abs

```
\def\BIC@Abs#1{%
  \ifx#1-%
    \expandafter\BIC@Space
  \else
    \expandafter\BIC@Space
    \expandafter#1%
  \fi
}
```

\bigintcalcSgn

```
\def\bigintcalcSgn#1{%
  \number
  \expandafter\expandafter\expandafter\BIC@Sgn
  \bigintcalcNum{#1}! %
}
```

\BIC@Sgn

```
\def\BIC@Sgn#1#2!{%
  \ifx#1-%
    -1%
  \else
    \ifx#10%
      0%
    \else
      1%
    \fi
  \fi
}
```

2.9 Cmp, Min, Max

\bigintcalcCmp

```
\def\bigintcalcCmp#1#2{%
  \number
  \expandafter\expandafter\expandafter\BIC@Cmp
  \bigintcalcNum{#2}!{#1}%
}
```

\BIC@Cmp

```
\def\BIC@Cmp#1!#2{%
  \expandafter\expandafter\expandafter\BIC@@Cmp
  \bigintcalcNum{#2}!#1!%
}
```

\BIC@@Cmp

```
\def\BIC@@Cmp#1#2!#3#4!{%
  \ifx#1-%
    \ifx#3-%
      \BIC@AfterFiFi{%
        \BIC@@Cmp#4!#2!%
      }%
    \else
      \BIC@AfterFiFi{%
        -1 %
      }%
    \fi
```

```
  \else
    \ifx#3-%
      \BIC@AfterFiFi{%
        1 %
      }%
    \else
      \BIC@AfterFiFi{%
        \BIC@CmpLength#1#2!#3#4!#1#2!#3#4!%
      }%
    \fi
  \BIC@Fi
}
```

`\BIC@PosCmp`

```
\def\BIC@PosCmp#1!#2!{%
  \BIC@CmpLength#1!#2!#1!#2!%
}
```

`\BIC@CmpLength`

```
\def\BIC@CmpLength#1#2!#3#4!{%
  \ifx\\#2\\%
    \ifx\\#4\\%
      \BIC@AfterFiFi\BIC@CmpDiff
    \else
      \BIC@AfterFiFi{%
        \BIC@CmpResult{-1}%
      }%
    \fi
  \else
    \ifx\\#4\\%
      \BIC@AfterFiFi{%
        \BIC@CmpResult1%
      }%
    \else
      \BIC@AfterFiFi{%
        \BIC@CmpLength#2!#4!%
      }%
    \fi
  \BIC@Fi
}
```

`\BIC@CmpResult`

```
\def\BIC@CmpResult#1#2!#3!{#1 }
```

`\BIC@CmpDiff`

```
\def\BIC@CmpDiff#1#2!#3#4!{%
  \ifnum#1<#3 %
    \BIC@AfterFi{%
      -1 %
    }%
  \else
    \ifnum#1>#3 %
      \BIC@AfterFiFi{%
        1 %
      }%
    \else
      \ifx\\#2\\%
        \BIC@AfterFiFiFi{%
```

```
          0 %
        }%
      \else
        \BIC@AfterFiFiFi{%
          \BIC@CmpDiff#2!#4!%
        }%
      \fi
    \fi
  \BIC@Fi
}
```

\bigintcalcMin

```
\def\bigintcalcMin#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@MinMax
  \bigintcalcNum{#1}!-!%
}
```

\bigintcalcMax

```
\def\bigintcalcMax#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@MinMax
  \bigintcalcNum{#1}!!%
}
```

\BIC@MinMax #1: x
#2: sign for comparison
#3: y

```
\def\BIC@MinMax#1!#2!#3{%
  \expandafter\expandafter\expandafter\BIC@@MinMax
  \bigintcalcNum{#3}!#1!#2!%
}
```

\BIC@@MinMax #1: y
#2: x
#3: sign for comparison

```
\def\BIC@@MinMax#1!#2!#3!{%
  \ifnum\BIC@@Cmp#1!#2!=#31 %
    \BIC@AfterFi{ #1}%
  \else
    \BIC@AfterFi{ #2}%
  \BIC@Fi
}
```

2.10 Odd

\bigintcalcOdd

```
\def\bigintcalcOdd#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Odd
  \bigintcalcAbs{#1}!%
}
```

\BigIntCalcOdd

```
\def\BigIntCalcOdd#1!{%
  \romannumeral0%
  \BIC@Odd#1!%
}
```

\BIC@Odd #1: x

```
\def\BIC@Odd#1#2{%
  \ifx#2!%
    \ifodd#1 %
      \BIC@AfterFiFi{ 1}%
    \else
      \BIC@AfterFiFi{ 0}%
    \fi
  \else
    \expandafter\BIC@Odd\expandafter#2%
  \BIC@Fi
}
```

2.11 Inc, Dec

\bigintcalcInc

```
\def\bigintcalcInc#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@IncSwitch
  \bigintcalcNum{#1}!%
}
```

\BIC@IncSwitch

```
\def\BIC@IncSwitch#1#2!{%
  \ifcase\BIC@@Cmp#1#2!-1!%
    \BIC@AfterFi{ 0}%
  \or
    \BIC@AfterFi{%
      \BIC@Inc#1#2!{}%
    }%
  \else
    \BIC@AfterFi{%
      \expandafter-\romannumeral0%
      \BIC@Dec#2!{}%
    }%
  \BIC@Fi
}
```

\bigintcalcDec

```
\def\bigintcalcDec#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@DecSwitch
  \bigintcalcNum{#1}!%
}
```

\BIC@DecSwitch

```
\def\BIC@DecSwitch#1#2!{%
  \ifcase\BIC@Sgn#1#2! %
    \BIC@AfterFi{ -1}%
  \or
    \BIC@AfterFi{%
      \BIC@Dec#1#2!{}%
    }%
  \else
    \BIC@AfterFi{%
      \expandafter-\romannumeral0%
      \BIC@Inc#2!{}%
```

```
    }%
  \BIC@Fi
}

\BigIntCalcInc
\def\BigIntCalcInc#1!{%
  \romannumeral0\BIC@Inc#1!{}%
}

\BigIntCalcDec
\def\BigIntCalcDec#1!{%
  \romannumeral0\BIC@Dec#1!{}%
}

\BIC@Inc
\def\BIC@Inc#1#2!#3{%
  \ifx\\#2\\%
    \BIC@AfterFi{%
      \BIC@@Inc1#1#3!{}%
    }%
  \else
    \BIC@AfterFi{%
      \BIC@Inc#2!{#1#3}%
    }%
  \BIC@Fi
}

\BIC@@Inc
\def\BIC@@Inc#1#2#3!#4{%
  \ifcase#1 %
    \ifx\\#3\\%
      \BIC@AfterFiFi{ #2#4}%
    \else
      \BIC@AfterFiFi{%
        \BIC@@Inc0#3!{#2#4}%
      }%
    \fi
  \else
    \ifnum#2<9 %
      \BIC@AfterFiFi{%
&       \expandafter\BIC@@@Inc\the\numexpr#2+1\relax
$       \expandafter\expandafter\expandafter\BIC@@@Inc
$       \ifcase#2 \expandafter1%
$       \or\expandafter2%
$       \or\expandafter3%
$       \or\expandafter4%
$       \or\expandafter5%
$       \or\expandafter6%
$       \or\expandafter7%
$       \or\expandafter8%
$       \or\expandafter9%
$?      \else\BigIntCalcError:ThisCannotHappen%
$       \fi
        0#3!{#4}%
      }%
    \else
      \BIC@AfterFiFi{%
        \BIC@@@Inc01#3!{#4}%
```

```
      }%
    \fi
  \BIC@Fi
}
```

\BIC@@@Inc

```
\def\BIC@@@Inc#1#2#3!#4{%
  \ifx\\#3\\%
    \ifnum#2=1 %
      \BIC@AfterFiFi{ 1#1#4}%
    \else
      \BIC@AfterFiFi{ #1#4}%
    \fi
  \else
    \BIC@AfterFi{%
      \BIC@@Inc#2#3!{#1#4}%
    }%
  \BIC@Fi
}
```

\BIC@Dec

```
\def\BIC@Dec#1#2!#3{%
  \ifx\\#2\\%
    \BIC@AfterFi{%
      \BIC@@Dec1#1#3!{}%
    }%
  \else
    \BIC@AfterFi{%
      \BIC@Dec#2!{#1#3}%
    }%
  \BIC@Fi
}
```

\BIC@@Dec

```
\def\BIC@@Dec#1#2#3!#4{%
  \ifcase#1 %
    \ifx\\#3\\%
      \BIC@AfterFiFi{ #2#4}%
    \else
      \BIC@AfterFiFi{%
        \BIC@@Dec0#3!{#2#4}%
      }%
    \fi
  \else
    \ifnum#2>0 %
      \BIC@AfterFiFi{%
&       \expandafter\BIC@@@Dec\the\numexpr#2-1\relax
$       \expandafter\expandafter\expandafter\BIC@@@Dec
$       \ifcase#2
$?        \BigIntCalcError:ThisCannotHappen%
$       \or\expandafter0%
$       \or\expandafter1%
$       \or\expandafter2%
$       \or\expandafter3%
$       \or\expandafter4%
$       \or\expandafter5%
$       \or\expandafter6%
$       \or\expandafter7%
```

```
$       \or\expandafter8%
$?      \else\BigIntCalcError:ThisCannotHappen%
$       \fi
        0#3!{#4}%
      }%
    \else
      \BIC@AfterFiFi{%
        \BIC@@@Dec91#3!{#4}%
      }%
    \fi
  \BIC@Fi
}
```

\BIC@@@Dec

```
\def\BIC@@@Dec#1#2#3!#4{%
  \ifx\\#3\\%
    \ifcase#1 %
      \ifx\\#4\\%
        \BIC@AfterFiFiFi{ 0}%
      \else
        \BIC@AfterFiFiFi{ #4}%
      \fi
    \else
      \BIC@AfterFiFi{ #1#4}%
    \fi
  \else
    \BIC@AfterFi{%
      \BIC@@Dec#2#3!{#1#4}%
    }%
  \BIC@Fi
}
```

2.12 Add, Sub

\bigintcalcAdd

```
\def\bigintcalcAdd#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Add
  \bigintcalcNum{#1}!%
}
```

\BIC@Add

```
\def\BIC@Add#1!#2{%
  \expandafter\expandafter\expandafter
  \BIC@AddSwitch\bigintcalcNum{#2}!#1!%
}
```

\bigintcalcSub

```
\def\bigintcalcSub#1#2{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Add
  \bigintcalcNum{-#2}!{#1}%
}
```

\BIC@AddSwitch Decision table for \BIC@AddSwitch.

$x < 0$	$y < 0$	$-x > -y$	$-$	$\mathrm{Add}(-x, -y)$
		else		$\mathrm{Add}(-y, -x)$
	else	$-x > y$	$-$	$\mathrm{Sub}(-x, y)$
		$-x = y$		0
		else	$+$	$\mathrm{Sub}(y, -x)$
else	$y < 0$	$x > -y$	$+$	$\mathrm{Sub}(x, -y)$
		$x = -y$		0
		else	$-$	$\mathrm{Sub}(-y, x)$
	else	$x > y$	$+$	$\mathrm{Add}(x, y)$
		else		$\mathrm{Add}(y, x)$

```
\def\BIC@AddSwitch#1#2!#3#4!{%
  \ifx#1-% x < 0
    \ifx#3-% y < 0
      \expandafter-\romannumeral0%
      \ifnum\BIC@PosCmp#2!#4!=1 % -x > -y
        \BIC@AfterFiFiFi{%
          \BIC@AddXY#2!#4!!!%
        }%
      \else % -x <= -y
        \BIC@AfterFiFiFi{%
          \BIC@AddXY#4!#2!!!%
        }%
      \fi
    \else % y >= 0
      \ifcase\BIC@PosCmp#2!#3#4!% -x = y
        \BIC@AfterFiFiFi{ 0}%
      \or % -x > y
        \expandafter-\romannumeral0%
        \BIC@AfterFiFiFi{%
          \BIC@SubXY#2!#3#4!!!%
        }%
      \else % -x <= y
        \BIC@AfterFiFiFi{%
          \BIC@SubXY#3#4!#2!!!%
        }%
      \fi
    \fi
  \else % x >= 0
    \ifx#3-% y < 0
      \ifcase\BIC@PosCmp#1#2!#4!% x = -y
        \BIC@AfterFiFiFi{ 0}%
      \or % x > -y
        \BIC@AfterFiFiFi{%
          \BIC@SubXY#1#2!#4!!!%
        }%
      \else % x <= -y
        \expandafter-\romannumeral0%
        \BIC@AfterFiFiFi{%
          \BIC@SubXY#4!#1#2!!!%
        }%
      \fi
    \else % y >= 0
      \ifnum\BIC@PosCmp#1#2!#3#4!=1 % x > y
        \BIC@AfterFiFiFi{%
          \BIC@AddXY#1#2!#3#4!!!%
        }%
```

```
      \else % x <= y
        \BIC@AfterFiFiFi{%
          \BIC@AddXY#3#4!#1#2!!!%
        }%
      \fi
    \fi
  \BIC@Fi
}
```

\BigIntCalcAdd

```
\def\BigIntCalcAdd#1!#2!{%
  \romannumeral0\BIC@AddXY#1!#2!!!%
}
```

\BigIntCalcSub

```
\def\BigIntCalcSub#1!#2!{%
  \romannumeral0\BIC@SubXY#1!#2!!!%
}
```

\BIC@AddXY

```
\def\BIC@AddXY#1#2!#3#4!#5!#6!{%
  \ifx\\#2\\%
    \ifx\\#3\\%
      \BIC@AfterFiFi{%
        \BIC@DoAdd0!#1#5!#60!%
      }%
    \else
      \BIC@AfterFiFi{%
        \BIC@DoAdd0!#1#5!#3#6!%
      }%
    \fi
  \else
    \ifx\\#4\\%
      \ifx\\#3\\%
        \BIC@AfterFiFiFi{%
          \BIC@AddXY#2!{}!#1#5!#60!%
        }%
      \else
        \BIC@AfterFiFiFi{%
          \BIC@AddXY#2!{}!#1#5!#3#6!%
        }%
      \fi
    \else
      \BIC@AfterFiFi{%
        \BIC@AddXY#2!#4!#1#5!#3#6!%
      }%
    \fi
  \BIC@Fi
}
```

\BIC@DoAdd #1: carry
#2: reverted result
#3#4: reverted x
#5#6: reverted y

```
\def\BIC@DoAdd#1#2!#3#4!#5#6!{%
  \ifx\\#4\\%
    \BIC@AfterFi{%
&     \expandafter\BIC@Space
```

```
&     \the\numexpr#1+#3+#5\relax#2%
$     \expandafter\expandafter\expandafter\BIC@AddResult
$     \BIC@AddDigit#1#3#5#2%
    }%
  \else
    \BIC@AfterFi{%
      \expandafter\expandafter\expandafter\BIC@DoAdd
      \BIC@AddDigit#1#3#5#2!#4!#6!%
    }%
  \BIC@Fi
}
```

`\BIC@AddResult`

```
$ \def\BIC@AddResult#1{%
$   \ifx#10%
$     \expandafter\BIC@Space
$   \else
$     \expandafter\BIC@Space\expandafter#1%
$   \fi
$ }%
```

`\BIC@AddDigit` #1: carry
#2: digit of x
#3: digit of y

```
\def\BIC@AddDigit#1#2#3{%
  \romannumeral0%
& \expandafter\BIC@@AddDigit\the\numexpr#1+#2+#3!%
$ \expandafter\BIC@@AddDigit\number%
$ \csname
$   BIC@AddCarry%
$   \ifcase#1 %
$     #2%
$   \else
$     \ifcase#2 1\or2\or3\or4\or5\or6\or7\or8\or9\or10\fi
$   \fi
$ \endcsname#3!%
}
```

`\BIC@@AddDigit`

```
\def\BIC@@AddDigit#1!{%
  \ifnum#1<10 %
    \BIC@AfterFi{ 0#1}%
  \else
    \BIC@AfterFi{ #1}%
  \BIC@Fi
}
```

`\BIC@AddCarry0`

```
$ \expandafter\def\csname BIC@AddCarry0\endcsname#1{#1}%
```

`\BIC@AddCarry10`

```
$ \expandafter\def\csname BIC@AddCarry10\endcsname#1{1#1}%
```

`\BIC@AddCarry[1-9]`

```
$ \def\BIC@Temp#1#2{%
$   \expandafter\def\csname BIC@AddCarry#1\endcsname##1{%
$     \ifcase##1 #1\or
```

```
$     #2%
$?    \else\BigIntCalcError:ThisCannotHappen%
$     \fi
$   }%
$ }%
$ \BIC@Temp 0{1\or2\or3\or4\or5\or6\or7\or8\or9}%
$ \BIC@Temp 1{2\or3\or4\or5\or6\or7\or8\or9\or10}%
$ \BIC@Temp 2{3\or4\or5\or6\or7\or8\or9\or10\or11}%
$ \BIC@Temp 3{4\or5\or6\or7\or8\or9\or10\or11\or12}%
$ \BIC@Temp 4{5\or6\or7\or8\or9\or10\or11\or12\or13}%
$ \BIC@Temp 5{6\or7\or8\or9\or10\or11\or12\or13\or14}%
$ \BIC@Temp 6{7\or8\or9\or10\or11\or12\or13\or14\or15}%
$ \BIC@Temp 7{8\or9\or10\or11\or12\or13\or14\or15\or16}%
$ \BIC@Temp 8{9\or10\or11\or12\or13\or14\or15\or16\or17}%
$ \BIC@Temp 9{10\or11\or12\or13\or14\or15\or16\or17\or18}%
```

`\BIC@SubXY` Preconditions:

- $x > y$, $x \geq 0$, and $y >= 0$
- $\text{digits}(x) = \text{digits}(y)$

```
\def\BIC@SubXY#1#2!#3#4!#5!#6!{%
  \ifx\\#2\\%
    \ifx\\#3\\%
      \BIC@AfterFiFi{%
        \BIC@DoSub0!#1#5!#60!%
      }%
    \else
      \BIC@AfterFiFi{%
        \BIC@DoSub0!#1#5!#3#6!%
      }%
    \fi
  \else
    \ifx\\#4\\%
      \ifx\\#3\\%
        \BIC@AfterFiFiFi{%
          \BIC@SubXY#2!{}!#1#5!#60!%
        }%
      \else
        \BIC@AfterFiFiFi{%
          \BIC@SubXY#2!{}!#1#5!#3#6!%
        }%
      \fi
    \else
      \BIC@AfterFiFi{%
        \BIC@SubXY#2!#4!#1#5!#3#6!%
      }%
    \fi
  \BIC@Fi
}
```

`\BIC@DoSub` #1: carry
#2: reverted result
#3#4: reverted x
#5#6: reverted y

```
\def\BIC@DoSub#1#2!#3#4!#5#6!{%
  \ifx\\#4\\%
    \BIC@AfterFi{%
```

```
      \expandafter\expandafter\expandafter\BIC@SubResult
      \BIC@SubDigit#1#3#5#2%
    }%
  \else
    \BIC@AfterFi{%
      \expandafter\expandafter\expandafter\BIC@DoSub
      \BIC@SubDigit#1#3#5#2!#4!#6!%
    }%
  \BIC@Fi
}
```

\BIC@SubResult

```
\def\BIC@SubResult#1{%
  \ifx#10%
    \expandafter\BIC@SubResult
  \else
    \expandafter\BIC@Space\expandafter#1%
  \fi
}
```

\BIC@SubDigit

#1: carry
#2: digit of x
#3: digit of y

```
\def\BIC@SubDigit#1#2#3{%
  \romannumeral0%
& \expandafter\BIC@@SubDigit\the\numexpr#2-#3-#1!%
$ \expandafter\BIC@@AddDigit\number
$   \csname
$     BIC@SubCarry%
$     \ifcase#1 %
$       #3%
$     \else
$       \ifcase#3 1\or2\or3\or4\or5\or6\or7\or8\or9\or10\fi
$     \fi
$   \endcsname#2!%
}
```

\BIC@@SubDigit

```
& \def\BIC@@SubDigit#1!{%
&   \ifnum#1<0 %
&     \BIC@AfterFi{%
&       \expandafter\BIC@Space
&       \expandafter1\the\numexpr#1+10\relax
&     }%
&   \else
&     \BIC@AfterFi{ 0#1}%
&   \BIC@Fi
& }%
```

\BIC@SubCarry0

```
$ \expandafter\def\csname BIC@SubCarry0\endcsname#1{#1}%
```

\BIC@SubCarry10

```
$ \expandafter\def\csname BIC@SubCarry10\endcsname#1{1#1}%
```

\BIC@SubCarry[1-9]

```
$ \def\BIC@Temp#1#2{%
```

```
$   \expandafter\def\csname BIC@SubCarry#1\endcsname##1{%
$     \ifcase##1 #2%
$?    \else\BigIntCalcError:ThisCannotHappen%
$     \fi
$   }%
$ }%
$ \BIC@Temp 1{19\or0\or1\or2\or3\or4\or5\or6\or7\or8}%
$ \BIC@Temp 2{18\or19\or0\or1\or2\or3\or4\or5\or6\or7}%
$ \BIC@Temp 3{17\or18\or19\or0\or1\or2\or3\or4\or5\or6}%
$ \BIC@Temp 4{16\or17\or18\or19\or0\or1\or2\or3\or4\or5}%
$ \BIC@Temp 5{15\or16\or17\or18\or19\or0\or1\or2\or3\or4}%
$ \BIC@Temp 6{14\or15\or16\or17\or18\or19\or0\or1\or2\or3}%
$ \BIC@Temp 7{13\or14\or15\or16\or17\or18\or19\or0\or1\or2}%
$ \BIC@Temp 8{12\or13\or14\or15\or16\or17\or18\or19\or0\or1}%
$ \BIC@Temp 9{11\or12\or13\or14\or15\or16\or17\or18\or19\or0}%
```

2.13 Shl, Shr

\bigintcalcShl

```
\def\bigintcalcShl#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Shl
  \bigintcalcNum{#1}!%
}
```

\BIC@Shl

```
\def\BIC@Shl#1#2!{%
  \ifx#1-%
    \BIC@AfterFi{%
      \expandafter-\romannumeral0%
&     \BIC@@Shl#2!!%
$     \BIC@AddXY#2!#2!!!%
    }%
  \else
    \BIC@AfterFi{%
&     \BIC@@Shl#1#2!!%
$     \BIC@AddXY#1#2!#1#2!!!%
    }%
  \BIC@Fi
}
```

\BigIntCalcShl

```
\def\BigIntCalcShl#1!{%
  \romannumeral0%
& \BIC@@Shl#1!!%
$ \BIC@AddXY#1!#1!!!%
}
```

\BIC@@Shl

```
& \def\BIC@@Shl#1#2!{%
&   \ifx\\#2\\%
&     \BIC@AfterFi{%
&       \BIC@@@Shl0!#1%
&     }%
&   \else
&     \BIC@AfterFi{%
&       \BIC@@Shl#2!#1%
```

```
&      }%
&    \BIC@Fi
& }%
```

`\BIC@@@Shl` #1: carry
#2: result
#3#4: reverted number

```
& \def\BIC@@@Shl#1#2!#3#4!{%
&   \ifx\\#4\\%
&     \BIC@AfterFi{%
&       \expandafter\BIC@Space
&       \the\numexpr#3*2+#1\relax#2%
&     }%
&   \else
&     \BIC@AfterFi{%
&       \expandafter\BIC@@@@Shl\the\numexpr#3*2+#1!#2!#4!%
&     }%
&   \BIC@Fi
& }%
```

`\BIC@@@@Shl`

```
& \def\BIC@@@@Shl#1!{%
&   \ifnum#1<10 %
&     \BIC@AfterFi{%
&       \BIC@@@Shl0#1%
&     }%
&   \else
&     \BIC@AfterFi{%
&       \BIC@@@Shl#1%
&     }%
&   \BIC@Fi
& }%
```

`\bigintcalcShr`

```
\def\bigintcalcShr#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Shr
  \bigintcalcNum{#1}!%
}
```

`\BIC@Shr`

```
\def\BIC@Shr#1#2!{%
  \ifx#1-%
    \expandafter-\romannumeral0%
    \BIC@AfterFi{%
      \BIC@@Shr#2!%
    }%
  \else
    \BIC@AfterFi{%
      \BIC@@Shr#1#2!%
    }%
  \BIC@Fi
}
```

`\BigIntCalcShr`

```
\def\BigIntCalcShr#1!{%
  \romannumeral0%
  \BIC@@Shr#1!%
}
```

\BIC@@Shr

```
\def\BIC@@Shr#1#2!{%
  \ifcase#1 %
    \BIC@AfterFi{ 0}%
  \or
    \ifx\\#2\\%
      \BIC@AfterFiFi{ 0}%
    \else
      \BIC@AfterFiFi{%
        \BIC@@@Shr#1#2!!%
      }%
    \fi
  \else
    \BIC@AfterFi{%
      \BIC@@@Shr0#1#2!!%
    }%
  \BIC@Fi
}
```

\BIC@@@Shr #1: carry
#2#3: number
#4: result

```
\def\BIC@@@Shr#1#2#3!#4!{%
  \ifx\\#3\\%
    \ifodd#1#2 %
      \BIC@AfterFiFi{%
&       \expandafter\BIC@ShrResult\the\numexpr(#1#2-1)/2\relax
$       \expandafter\expandafter\expandafter\BIC@ShrResult
$       \csname BIC@ShrDigit#1#2\endcsname
        #4!%
      }%
    \else
      \BIC@AfterFiFi{%
&       \expandafter\BIC@ShrResult\the\numexpr#1#2/2\relax
$       \expandafter\expandafter\expandafter\BIC@ShrResult
$       \csname BIC@ShrDigit#1#2\endcsname
        #4!%
      }%
    \fi
  \else
    \ifodd#1#2 %
      \BIC@AfterFiFi{%
&       \expandafter\BIC@@@@Shr\the\numexpr(#1#2-1)/2\relax1%
$       \expandafter\expandafter\expandafter\BIC@@@@Shr
$       \csname BIC@ShrDigit#1#2\endcsname
        #3!#4!%
      }%
    \else
      \BIC@AfterFiFi{%
&       \expandafter\BIC@@@@Shr\the\numexpr#1#2/2\relax0%
$       \expandafter\expandafter\expandafter\BIC@@@@Shr
$       \csname BIC@ShrDigit#1#2\endcsname
        #3!#4!%
      }%
    \fi
  \BIC@Fi
}
```

\BIC@ShrResult

```
& \def\BIC@ShrResult#1#2!{ #2#1}%
$ \def\BIC@ShrResult#1#2#3!{ #3#1}%
```

\BIC@@@@Shr

#1: new digit
#2: carry
#3: remaining number
#4: result

```
\def\BIC@@@@Shr#1#2#3!#4!{%
  \BIC@@@Shr#2#3!#4#1!%
}
```

\BIC@ShrDigit[00-19]

```
$ \def\BIC@Temp#1#2#3#4{%
$   \expandafter\def\csname BIC@ShrDigit#1#2\endcsname{#3#4}%
$ }%
$ \BIC@Temp 0000%
$ \BIC@Temp 0101%
$ \BIC@Temp 0210%
$ \BIC@Temp 0311%
$ \BIC@Temp 0420%
$ \BIC@Temp 0521%
$ \BIC@Temp 0630%
$ \BIC@Temp 0731%
$ \BIC@Temp 0840%
$ \BIC@Temp 0941%
$ \BIC@Temp 1050%
$ \BIC@Temp 1151%
$ \BIC@Temp 1260%
$ \BIC@Temp 1361%
$ \BIC@Temp 1470%
$ \BIC@Temp 1571%
$ \BIC@Temp 1680%
$ \BIC@Temp 1781%
$ \BIC@Temp 1890%
$ \BIC@Temp 1991%
```

2.14 \BIC@Tim

\BIC@Tim

Macro \BIC@Tim implements "Number *tim*es digit".
#1: plain number without sign
#2: digit

\BIC@@Tim

#1#2: number
#3: reverted number

```
\def\BIC@@Tim#1#2!{%
  \ifx\\#2\\%
    \BIC@AfterFi{%
      \BIC@ProcessTim0!#1%
    }%
  \else
    \BIC@AfterFi{%
      \BIC@@Tim#2!#1%
    }%
  \BIC@Fi
}
```

`\BIC@ProcessTim` #1: carry
#2: result
#3#4: reverted number
#5: digit

```
\def\BIC@ProcessTim#1#2!#3#4!#5{%
  \ifx\\#4\\%
    \BIC@AfterFi{%
      \expandafter\BIC@Space
&     \the\numexpr#3*#5+#1\relax
$     \romannumeral0\BIC@TimDigit#3#5#1%
      #2%
    }%
  \else
    \BIC@AfterFi{%
      \expandafter\BIC@@ProcessTim
&     \the\numexpr#3*#5+#1%
$     \romannumeral0\BIC@TimDigit#3#5#1%
      !#2!#4!#5%
    }%
  \BIC@Fi
}
```

`\BIC@@ProcessTim` #1#2: carry?, new digit
#3: new number
#4: old number
#5: digit

```
\def\BIC@@ProcessTim#1#2!{%
  \ifx\\#2\\%
    \BIC@AfterFi{%
      \BIC@ProcessTim0#1%
    }%
  \else
    \BIC@AfterFi{%
      \BIC@ProcessTim#1#2%
    }%
  \BIC@Fi
}
```

`\BIC@TimDigit` #1: digit 0–9
#2: digit 3–9
#3: carry 0–9

```
$ \def\BIC@TimDigit#1#2#3{%
$   \ifcase#1 % 0
$     \BIC@AfterFi{ #3}%
$   \or % 1
$     \BIC@AfterFi{%
$       \expandafter\BIC@Space
$       \number\csname BIC@AddCarry#2\endcsname#3 %
$     }%
$   \else
$     \ifcase#3 %
$       \BIC@AfterFiFi{%
$         \expandafter\BIC@Space
$         \number\csname BIC@MulDigit#2\endcsname#1 %
$       }%
$     \else
$       \BIC@AfterFiFi{%
```

```
$         \expandafter\BIC@Space
$         \romannumeral0%
$         \expandafter\BIC@AddXY
$         \number\csname BIC@MulDigit#2\endcsname#1!%
$         #3!!!%
$       }%
$     \fi
$   \BIC@Fi
$ }%
```

`\BIC@MulDigit[3-9]`

```
$ \def\BIC@Temp#1#2{%
$   \expandafter\def\csname BIC@MulDigit#1\endcsname##1{%
$     \ifcase##1 0%
$     \or ##1%
$     \or #2%
$?    \else\BigIntCalcError:ThisCannotHappen%
$     \fi
$   }%
$ }%
$ \BIC@Temp 3{6\or9\or12\or15\or18\or21\or24\or27}%
$ \BIC@Temp 4{8\or12\or16\or20\or24\or28\or32\or36}%
$ \BIC@Temp 5{10\or15\or20\or25\or30\or35\or40\or45}%
$ \BIC@Temp 6{12\or18\or24\or30\or36\or42\or48\or54}%
$ \BIC@Temp 7{14\or21\or28\or35\or42\or49\or56\or63}%
$ \BIC@Temp 8{16\or24\or32\or40\or48\or56\or64\or72}%
$ \BIC@Temp 9{18\or27\or36\or45\or54\or63\or72\or81}%
```

2.15 Mul

`\bigintcalcMul`

```
\def\bigintcalcMul#1#2{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Mul
  \bigintcalcNum{#1}!{#2}%
}
```

`\BIC@Mul`

```
\def\BIC@Mul#1!#2{%
  \expandafter\expandafter\expandafter\BIC@MulSwitch
  \bigintcalcNum{#2}!#1!%
}
```

`\BIC@MulSwitch` Decision table for `\BIC@MulSwitch`.

$x = 0$				0
$x > 0$	$y = 0$			0
	$y > 0$	$x > y$	$+$	$\mathrm{Mul}(x, y)$
		else		$\mathrm{Mul}(y, x)$
	$y < 0$	$x > -y$	$-$	$\mathrm{Mul}(x, -y)$
		else		$\mathrm{Mul}(-y, x)$
$x < 0$	$y = 0$			0
	$y > 0$	$-x > y$	$-$	$\mathrm{Mul}(-x, y)$
		else		$\mathrm{Mul}(y, -x)$
	$y < 0$	$-x > -y$	$+$	$\mathrm{Mul}(-x, -y)$
		else		$\mathrm{Mul}(-y, -x)$

```
\def\BIC@MulSwitch#1#2!#3#4!{%
  \ifcase\BIC@Sgn#1#2! % x = 0
    \BIC@AfterFi{ 0}%
  \or % x > 0
    \ifcase\BIC@Sgn#3#4! % y = 0
      \BIC@AfterFiFi{ 0}%
    \or % y > 0
      \ifnum\BIC@PosCmp#1#2!#3#4!=1 % x > y
        \BIC@AfterFiFiFi{%
          \BIC@ProcessMul0!#1#2!#3#4!%
        }%
      \else % x <= y
        \BIC@AfterFiFiFi{%
          \BIC@ProcessMul0!#3#4!#1#2!%
        }%
      \fi
    \else % y < 0
      \expandafter-\romannumeral0%
      \ifnum\BIC@PosCmp#1#2!#4!=1 % x > -y
        \BIC@AfterFiFiFi{%
          \BIC@ProcessMul0!#1#2!#4!%
        }%
      \else % x <= -y
        \BIC@AfterFiFiFi{%
          \BIC@ProcessMul0!#4!#1#2!%
        }%
      \fi
    \fi
  \else % x < 0
    \ifcase\BIC@Sgn#3#4! % y = 0
      \BIC@AfterFiFi{ 0}%
    \or % y > 0
      \expandafter-\romannumeral0%
      \ifnum\BIC@PosCmp#2!#3#4!=1 % -x > y
        \BIC@AfterFiFiFi{%
          \BIC@ProcessMul0!#2!#3#4!%
        }%
      \else % -x <= y
        \BIC@AfterFiFiFi{%
          \BIC@ProcessMul0!#3#4!#2!%
        }%
      \fi
    \else % y < 0
      \ifnum\BIC@PosCmp#2!#4!=1 % -x > -y
        \BIC@AfterFiFiFi{%
          \BIC@ProcessMul0!#2!#4!%
        }%
      \else % -x <= -y
        \BIC@AfterFiFiFi{%
          \BIC@ProcessMul0!#4!#2!%
        }%
      \fi
    \fi
  \BIC@Fi
}
```

\BigIntCalcMul

```
\def\BigIntCalcMul#1!#2!{%
```

```
  \romannumeral0%
  \BIC@ProcessMul0!#1!#2!%
}
```

`\BIC@ProcessMul` #1: result
#2: number x
#3#4: number y

```
\def\BIC@ProcessMul#1!#2!#3#4!{%
  \ifx\\#4\\%
    \BIC@AfterFi{%
      \expandafter\expandafter\expandafter\BIC@Space
      \bigintcalcAdd{\BIC@Tim#2!#3}{#10}%
    }%
  \else
    \BIC@AfterFi{%
      \expandafter\expandafter\expandafter\BIC@ProcessMul
      \bigintcalcAdd{\BIC@Tim#2!#3}{#10}!#2!#4!%
    }%
  \BIC@Fi
}
```

2.16 Sqr

`\bigintcalcSqr`

```
\def\bigintcalcSqr#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Sqr
  \bigintcalcNum{#1}!%
}
```

`\BIC@Sqr`

```
\def\BIC@Sqr#1{%
   \ifx#1-%
     \expandafter\BIC@@Sqr
   \else
     \expandafter\BIC@@Sqr\expandafter#1%
   \fi
}
```

`\BIC@@Sqr`

```
\def\BIC@@Sqr#1!{%
  \BIC@ProcessMul0!#1!#1!%
}
```

2.17 Fac

`\bigintcalcFac`

```
\def\bigintcalcFac#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Fac
  \bigintcalcNum{#1}!%
}
```

`\BIC@Fac`

```
\def\BIC@Fac#1#2!{%
  \ifx#1-%
    \BIC@AfterFi{ 0\BigIntCalcError:FacNegative}%
```

```
  \else
    \ifnum\BIC@PosCmp#1#2!13!<0 %
      \ifcase#1#2 %
         \BIC@AfterFiFiFi{ 1}% 0!
      \or\BIC@AfterFiFiFi{ 1}% 1!
      \or\BIC@AfterFiFiFi{ 2}% 2!
      \or\BIC@AfterFiFiFi{ 6}% 3!
      \or\BIC@AfterFiFiFi{ 24}% 4!
      \or\BIC@AfterFiFiFi{ 120}% 5!
      \or\BIC@AfterFiFiFi{ 720}% 6!
      \or\BIC@AfterFiFiFi{ 5040}% 7!
      \or\BIC@AfterFiFiFi{ 40320}% 8!
      \or\BIC@AfterFiFiFi{ 362880}% 9!
      \or\BIC@AfterFiFiFi{ 3628800}% 10!
      \or\BIC@AfterFiFiFi{ 39916800}% 11!
      \or\BIC@AfterFiFiFi{ 479001600}% 12!
?     \else\BigIntCalcError:ThisCannotHappen%
      \fi
    \else
      \BIC@AfterFiFi{%
        \BIC@ProcessFac#1#2!479001600!%
      }%
    \fi
  \BIC@Fi
}
```

`\BIC@ProcessFac` #1: n
#2: result

```
\def\BIC@ProcessFac#1!#2!{%
  \ifnum\BIC@PosCmp#1!12!=0 %
    \BIC@AfterFi{ #2}%
  \else
    \BIC@AfterFi{%
      \expandafter\BIC@@ProcessFac
      \romannumeral0\BIC@ProcessMul0!#2!#1!%
      !#1!%
    }%
  \BIC@Fi
}
```

`\BIC@@ProcessFac` #1: result
#2: n

```
\def\BIC@@ProcessFac#1!#2!{%
  \expandafter\BIC@ProcessFac
  \romannumeral0\BIC@Dec#2!{}%
  !#1!%
}
```

2.18 Pow

`\bigintcalcPow` #1: basis
#2: power

```
\def\bigintcalcPow#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Pow
  \bigintcalcNum{#1}!%
}
```

`\BIC@Pow` `#1`: basis
`#2`: power

```
\def\BIC@Pow#1!#2{%
  \expandafter\expandafter\expandafter\BIC@PowSwitch
  \bigintcalcNum{#2}!#1!%
}
```

`\BIC@PowSwitch` `#1#2`: power y
`#3#4`: basis x
Decision table for `\BIC@PowSwitch`.

$y = 0$			1
$y = 1$			x
$y = 2$	$x < 0$		$\mathrm{Mul}(-x, -x)$
	else		$\mathrm{Mul}(x, x)$
$y < 0$	$x = 0$		DivisionByZero
	$x = 1$		1
	$x = -1$	$\mathrm{ifodd}(y)$	-1
		else	1
	else ($\lvert x\rvert > 1$)		0
$y > 2$	$x = 0$		0
	$x = 1$		1
	$x = -1$	$\mathrm{ifodd}(y)$	-1
		else	1
	$x < -1$ ($x < 0$)	$\mathrm{ifodd}(y)$	$-\mathrm{Pow}(-x, y)$
		else	$\mathrm{Pow}(-x, y)$
	else ($x > 1$)		$\mathrm{Pow}(x, y)$

```
\def\BIC@PowSwitch#1#2!#3#4!{%
  \ifcase\ifx\\#2\\%
           \ifx#100 % y = 0
           \else\ifx#111 % y = 1
           \else\ifx#122 % y = 2
           \else4 % y > 2
           \fi\fi\fi
         \else
           \ifx#1-3 % y < 0
           \else4 % y > 2
           \fi
         \fi
    \BIC@AfterFi{ 1}% y = 0
  \or % y = 1
    \BIC@AfterFi{ #3#4}%
  \or % y = 2
    \ifx#3-% x < 0
      \BIC@AfterFiFi{%
        \BIC@ProcessMul0!#4!#4!%
      }%
    \else % x >= 0
      \BIC@AfterFiFi{%
        \BIC@ProcessMul0!#3#4!#3#4!%
      }%
    \fi
  \or % y < 0
    \ifcase\ifx\\#4\\%
             \ifx#300 % x = 0
```

```
            \else\ifx#311 % x = 1
            \else3 % x > 1
            \fi\fi
          \else
            \ifcase\BIC@MinusOne#3#4! %
              3 % |x| > 1
            \or
              2 % x = -1
            \else\BigIntCalcError:ThisCannotHappen%
            \fi
          \fi
      \BIC@AfterFiFi{ 0\BigIntCalcError:DivisionByZero}% x = 0
    \or % x = 1
      \BIC@AfterFiFi{ 1}% x = 1
    \or % x = -1
      \ifcase\BIC@ModTwo#2! % even(y)
        \BIC@AfterFiFiFi{ 1}%
      \or % odd(y)
        \BIC@AfterFiFiFi{ -1}%
    \else\BigIntCalcError:ThisCannotHappen%
      \fi
    \or % |x| > 1
      \BIC@AfterFiFi{ 0}%
  \else\BigIntCalcError:ThisCannotHappen%
    \fi
  \or % y > 2
    \ifcase\ifx\\#4\\%
            \ifx#300 % x = 0
            \else\ifx#311 % x = 1
            \else4 % x > 1
            \fi\fi
          \else
            \ifx#3-%
              \ifcase\BIC@MinusOne#3#4! %
                3 % x < -1
              \else
                2 % x = -1
              \fi
            \else
              4 % x > 1
            \fi
          \fi
      \BIC@AfterFiFi{ 0}% x = 0
    \or % x = 1
      \BIC@AfterFiFi{ 1}% x = 1
    \or % x = -1
      \ifcase\BIC@ModTwo#1#2! % even(y)
        \BIC@AfterFiFiFi{ 1}%
      \or % odd(y)
        \BIC@AfterFiFiFi{ -1}%
    \else\BigIntCalcError:ThisCannotHappen%
      \fi
    \or % x < -1
      \ifcase\BIC@ModTwo#1#2! % even(y)
        \BIC@AfterFiFiFi{%
          \BIC@PowRec#4!#1#2!1!%
        }%
      \or % odd(y)
```

```
        \expandafter-\romannumeral0%
        \BIC@AfterFiFiFi{%
          \BIC@PowRec#4!#1#2!1!%
        }%
?     \else\BigIntCalcError:ThisCannotHappen%
      \fi
    \or % x > 1
      \BIC@AfterFiFi{%
        \BIC@PowRec#3#4!#1#2!1!%
      }%
?   \else\BigIntCalcError:ThisCannotHappen%
    \fi
? \else\BigIntCalcError:ThisCannotHappen%
  \BIC@Fi
}
```

2.18.1 Help macros

`\BIC@ModTwo` Macro `\BIC@ModTwo` expects a number without sign and returns digit `1` or `0` if the number is odd or even.

```
\def\BIC@ModTwo#1#2!{%
  \ifx\\#2\\%
    \ifodd#1 %
      \BIC@AfterFiFi1%
    \else
      \BIC@AfterFiFi0%
    \fi
  \else
    \BIC@AfterFi{%
      \BIC@ModTwo#2!%
    }%
  \BIC@Fi
}
```

`\BIC@MinusOne` Macro `\BIC@MinusOne` expects a number and returns digit `1` if the number equals minus one and returns `0` otherwise.

```
\def\BIC@MinusOne#1#2!{%
  \ifx#1-%
    \BIC@@MinusOne#2!%
  \else
    0%
  \fi
}
```

`\BIC@@MinusOne`

```
\def\BIC@@MinusOne#1#2!{%
  \ifx#11%
    \ifx\\#2\\%
      1%
    \else
      0%
    \fi
  \else
    0%
  \fi
}
```

2.18.2 Recursive calculation

\BIC@PowRec

```
Pow(x, y) {
  PowRec(x, y, 1)
}
PowRec(x, y, r) {
  if y == 1 then
    return r
  else
    ifodd y then
      return PowRec(x*x, y div 2, r*x) % y div 2 = (y-1)/2
    else
      return PowRec(x*x, y div 2, r)
    fi
  fi
}
```

#1: x (basis)
#2#3: y (power)
#4: r (result)

```
\def\BIC@PowRec#1!#2#3!#4!{%
 \ifcase\ifx#21\ifx\\#3\\0 \else1 \fi\else1 \fi % y = 1
  \ifnum\BIC@PosCmp#1!#4!=1 % x > r
    \BIC@AfterFiFi{%
      \BIC@ProcessMul0!#1!#4!%
    }%
  \else
    \BIC@AfterFiFi{%
      \BIC@ProcessMul0!#4!#1!%
    }%
  \fi
 \or
  \ifcase\BIC@ModTwo#2#3! % even(y)
    \BIC@AfterFiFi{%
      \expandafter\BIC@@PowRec\romannumeral0%
      \BIC@@Shr#2#3!%
      !#1!#4!%
    }%
  \or % odd(y)
    \ifnum\BIC@PosCmp#1!#4!=1 % x > r
      \BIC@AfterFiFiFi{%
        \expandafter\BIC@@@PowRec\romannumeral0%
        \BIC@ProcessMul0!#1!#4!%
        !#1!#2#3!%
      }%
    \else
      \BIC@AfterFiFiFi{%
        \expandafter\BIC@@@PowRec\romannumeral0%
        \BIC@ProcessMul0!#1!#4!%
        !#1!#2#3!%
      }%
    \fi
? \else\BigIntCalcError:ThisCannotHappen%
  \fi
? \else\BigIntCalcError:ThisCannotHappen%
 \BIC@Fi
}
```

\BIC@@PowRec

#1: $y/2$

#2: x
#3: new r (r or $r * x$)

```
\def\BIC@@PowRec#1!#2!#3!{%
  \expandafter\BIC@PowRec\romannumeral0%
  \BIC@ProcessMul0!#2!#2!%
  !#1!#3!%
}
```

`\BIC@@@PowRec` #1: $r * x$ #2: x #3: y

```
\def\BIC@@@PowRec#1!#2!#3!{%
  \expandafter\BIC@@PowRec\romannumeral0%
  \BIC@@Shr#3!%
  !#2!#1!%
}
```

2.19 Div

`\bigintcalcDiv` #1: x
#2: y (divisor)

```
\def\bigintcalcDiv#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Div
  \bigintcalcNum{#1}!%
}
```

`\BIC@Div` #1: x
#2: y

```
\def\BIC@Div#1!#2{%
  \expandafter\expandafter\expandafter\BIC@DivSwitchSign
  \bigintcalcNum{#2}!#1!%
}
```

`\BigIntCalcDiv`

```
\def\BigIntCalcDiv#1!#2!{%
  \romannumeral0%
  \BIC@DivSwitchSign#2!#1!%
}
```

`\BIC@DivSwitchSign` Decision table for `\BIC@DivSwitchSign`.

$y = 0$	DivisionByZero	
$y > 0$	$x = 0$	0
	$x > 0$	DivSwitch$(+, x, y)$
	$x < 0$	DivSwitch$(-, -x, y)$
$y < 0$	$x = 0$	0
	$x > 0$	DivSwitch$(-, x, -y)$
	$x < 0$	DivSwitch$(+, -x, -y)$

#1: y (divisor)
#2: x

```
\def\BIC@DivSwitchSign#1#2!#3#4!{%
  \ifcase\BIC@Sgn#1#2! % y = 0
    \BIC@AfterFi{ 0\BigIntCalcError:DivisionByZero}%
  \or % y > 0
    \ifcase\BIC@Sgn#3#4! % x = 0
      \BIC@AfterFiFi{ 0}%
```

```
    \or % x > 0
      \BIC@AfterFiFi{%
        \BIC@DivSwitch{}#3#4!#1#2!%
      }%
    \else % x < 0
      \BIC@AfterFiFi{%
        \BIC@DivSwitch-#4!#1#2!%
      }%
    \fi
  \else % y < 0
    \ifcase\BIC@Sgn#3#4! % x = 0
      \BIC@AfterFiFi{ 0}%
    \or % x > 0
      \BIC@AfterFiFi{%
        \BIC@DivSwitch-#3#4!#2!%
      }%
    \else % x < 0
      \BIC@AfterFiFi{%
        \BIC@DivSwitch{}#4!#2!%
      }%
    \fi
  \BIC@Fi
}
```

`\BIC@DivSwitch` Decision table for `\BIC@DivSwitch`.

$y = x$	sign 1	
$y > x$	0	
$y < x$	$y = 1$	sign x
	$y = 2$	sign $\mathrm{Shr}(x)$
	$y = 4$	sign $\mathrm{Shr}(\mathrm{Shr}(x))$
	else	sign $\mathrm{ProcessDiv}(x, y)$

#1: sign
#2: x
#3#4: y ($y \neq 0$)

```
\def\BIC@DivSwitch#1#2!#3#4!{%
  \ifcase\BIC@PosCmp#3#4!#2!% y = x
    \BIC@AfterFi{ #11}%
  \or % y > x
    \BIC@AfterFi{ 0}%
  \else % y < x
    \ifx\\#1\\%
    \else
      \expandafter-\romannumeral0%
    \fi
    \ifcase\ifx\\#4\\%
             \ifx#310 % y = 1
             \else\ifx#321 % y = 2
             \else\ifx#342 % y = 4
             \else3 % y > 2
             \fi\fi\fi
           \else
             3 % y > 2
           \fi
      \BIC@AfterFiFi{ #2}% y = 1
    \or % y = 2
      \BIC@AfterFiFi{%
```

```
        \BIC@@Shr#2!%
      }%
    \or % y = 4
      \BIC@AfterFiFi{%
        \expandafter\BIC@@Shr\romannumeral0%
          \BIC@@Shr#2!!%
      }%
    \or % y > 2
      \BIC@AfterFiFi{%
        \BIC@DivStartX#2!#3#4!!!%
      }%
?   \else\BigIntCalcError:ThisCannotHappen%
    \fi
  \BIC@Fi
}
```

`\BIC@ProcessDiv` #1#2: x
#3#4: y
#5: collect first digits of x
#6: corresponding digits of y

```
\def\BIC@DivStartX#1#2!#3#4!#5!#6!{%
  \ifx\\#4\\%
    \BIC@AfterFi{%
      \BIC@DivStartYii#6#3#4!{#5#1}#2=!%
    }%
  \else
    \BIC@AfterFi{%
      \BIC@DivStartX#2!#4!#5#1!#6#3!%
    }%
  \BIC@Fi
}
```

`\BIC@DivStartYii` #1: y
#2: x, =

```
\def\BIC@DivStartYii#1!{%
  \expandafter\BIC@DivStartYiv\romannumeral0%
  \BIC@Shl#1!%
  !#1!%
}
```

`\BIC@DivStartYiv` #1: $2y$
#2: y
#3: x, =

```
\def\BIC@DivStartYiv#1!{%
  \expandafter\BIC@DivStartYvi\romannumeral0%
  \BIC@Shl#1!%
  !#1!%
}
```

`\BIC@DivStartYvi` #1: $4y$
#2: $2y$
#3: y
#4: x, =

```
\def\BIC@DivStartYvi#1!#2!{%
  \expandafter\BIC@DivStartYviii\romannumeral0%
  \BIC@AddXY#1!#2!!!%
  !#1!#2!%
}
```

\BIC@DivStartYviii #1: $6y$
#2: $4y$
#3: $2y$
#4: y
#5: x, =

```
\def\BIC@DivStartYviii#1!#2!{%
  \expandafter\BIC@DivStart\romannumeral0%
  \BIC@Shl#2!%
  !#1!#2!%
}
```

\BIC@DivStart #1: $8y$
#2: $6y$
#3: $4y$
#4: $2y$
#5: y
#6: x, =

```
\def\BIC@DivStart#1!#2!#3!#4!#5!#6!{%
  \BIC@ProcessDiv#6!!#5!#4!#3!#2!#1!=%
}
```

\BIC@ProcessDiv #1#2#3: x, =
#4: result
#5: y
#6: $2y$
#7: $4y$
#8: $6y$
#9: $8y$

```
\def\BIC@ProcessDiv#1#2#3!#4!#5!{%
  \ifcase\BIC@PosCmp#5!#1!% y = #1
    \ifx#2=%
      \BIC@AfterFiFi{\BIC@DivCleanup{#41}}%
    \else
      \BIC@AfterFiFi{%
        \BIC@ProcessDiv#2#3!#41!#5!%
      }%
    \fi
  \or % y > #1
    \ifx#2=%
      \BIC@AfterFiFi{\BIC@DivCleanup{#40}}%
    \else
      \ifx\\#4\\%
        \BIC@AfterFiFiFi{%
          \BIC@ProcessDiv{#1#2}#3!!#5!%
        }%
      \else
        \BIC@AfterFiFiFi{%
          \BIC@ProcessDiv{#1#2}#3!#40!#5!%
        }%
      \fi
    \fi
  \else % y < #1
    \BIC@AfterFi{%
      \BIC@@ProcessDiv{#1}#2#3!#4!#5!%
    }%
  \BIC@Fi
}
```

\BIC@DivCleanup #1: result
#2: garbage

```
\def\BIC@DivCleanup#1#2={ #1}%
```

\BIC@@ProcessDiv

```
\def\BIC@@ProcessDiv#1#2#3!#4!#5!#6!#7!{%
  \ifcase\BIC@PosCmp#7!#1!% 4y = #1
    \ifx#2=%
      \BIC@AfterFiFi{\BIC@DivCleanup{#44}}%
    \else
     \BIC@AfterFiFi{%
       \BIC@ProcessDiv#2#3!#44!#5!#6!#7!%
     }%
    \fi
  \or % 4y > #1
    \ifcase\BIC@PosCmp#6!#1!% 2y = #1
      \ifx#2=%
        \BIC@AfterFiFiFi{\BIC@DivCleanup{#42}}%
      \else
        \BIC@AfterFiFiFi{%
          \BIC@ProcessDiv#2#3!#42!#5!#6!#7!%
        }%
      \fi
    \or % 2y > #1
      \ifx#2=%
        \BIC@AfterFiFiFi{\BIC@DivCleanup{#41}}%
      \else
        \BIC@AfterFiFiFi{%
          \BIC@DivSub#1!#5!#2#3!#41!#5!#6!#7!%
        }%
      \fi
    \else % 2y < #1
      \BIC@AfterFiFi{%
        \expandafter\BIC@ProcessDivII\romannumeral0%
        \BIC@SubXY#1!#6!!!%
        !#2#3!#4!#5!23%
        #6!#7!%
      }%
    \fi
  \else % 4y < #1
    \BIC@AfterFi{%
      \BIC@@@ProcessDiv{#1}#2#3!#4!#5!#6!#7!%
    }%
  \BIC@Fi
}
```

\BIC@DivSub Next token group: #1-#2 and next digit #3.

```
\def\BIC@DivSub#1!#2!#3{%
  \expandafter\BIC@ProcessDiv\expandafter{%
    \romannumeral0%
    \BIC@SubXY#1!#2!!!%
    #3%
  }%
}
```

\BIC@ProcessDivII #1: $x' - 2y$
#2#3: remaining x, =
#4: result

#5: y
#6: first possible result digit
#7: second possible result digit

```
\def\BIC@ProcessDivII#1!#2#3!#4!#5!#6#7{%
  \ifcase\BIC@PosCmp#5!#1!% y = #1
    \ifx#2=%
      \BIC@AfterFiFi{\BIC@DivCleanup{#4#7}}%
    \else
      \BIC@AfterFiFi{%
        \BIC@ProcessDiv#2#3!#4#7!#5!%
      }%
    \fi
  \or % y > #1
    \ifx#2=%
      \BIC@AfterFiFi{\BIC@DivCleanup{#4#6}}%
    \else
      \BIC@AfterFiFi{%
        \BIC@ProcessDiv{#1#2}#3!#4#6!#5!%
      }%
    \fi
  \else % y < #1
    \ifx#2=%
      \BIC@AfterFiFi{\BIC@DivCleanup{#4#7}}%
    \else
      \BIC@AfterFiFi{%
        \BIC@DivSub#1!#5!#2#3!#4#7!#5!%
      }%
    \fi
  \BIC@Fi
}
```

`\BIC@ProcessDivIV`

#1#2#3: x, =, $x > 4y$
#4: result
#5: y
#6: $2y$
#7: $4y$
#8: $6y$
#9: $8y$

```
\def\BIC@@@ProcessDiv#1#2#3!#4!#5!#6!#7!#8!#9!{%
  \ifcase\BIC@PosCmp#8!#1!% 6y = #1
    \ifx#2=%
      \BIC@AfterFiFi{\BIC@DivCleanup{#46}}%
    \else
      \BIC@AfterFiFi{%
        \BIC@ProcessDiv#2#3!#46!#5!#6!#7!#8!#9!%
      }%
    \fi
  \or % 6y > #1
    \BIC@AfterFi{%
      \expandafter\BIC@ProcessDivII\romannumeral0%
      \BIC@SubXY#1!#7!!!%
      !#2#3!#4!#5!45%
      #6!#7!#8!#9!%
    }%
  \else % 6y < #1
    \ifcase\BIC@PosCmp#9!#1!% 8y = #1
      \ifx#2=%
```

```
        \BIC@AfterFiFiFi{\BIC@DivCleanup{#48}}%
      \else
        \BIC@AfterFiFiFi{%
          \BIC@ProcessDiv#2#3!#48!#5!#6!#7!#8!#9!%
        }%
      \fi
    \or % 8y > #1
      \BIC@AfterFiFi{%
        \expandafter\BIC@ProcessDivII\romannumeral0%
        \BIC@SubXY#1!#8!!!%
        !#2#3!#4!#5!67%
        #6!#7!#8!#9!%
      }%
    \else % 8y < #1
      \BIC@AfterFiFi{%
        \expandafter\BIC@ProcessDivII\romannumeral0%
        \BIC@SubXY#1!#9!!!%
        !#2#3!#4!#5!89%
        #6!#7!#8!#9!%
      }%
    \fi
  \BIC@Fi
}
```

2.20 Mod

\bigintcalcMod

#1: x
#2: y

```
\def\bigintcalcMod#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\BIC@Mod
  \bigintcalcNum{#1}!%
}
```

\BIC@Mod

#1: x
#2: y

```
\def\BIC@Mod#1!#2{%
  \expandafter\expandafter\expandafter\BIC@ModSwitchSign
  \bigintcalcNum{#2}!#1!%
}
```

\BigIntCalcMod

```
\def\BigIntCalcMod#1!#2!{%
  \romannumeral0%
  \BIC@ModSwitchSign#2!#1!%
}
```

\BIC@ModSwitchSign

Decision table for \BIC@ModSwitchSign.

$y = 0$	DivisionByZero	
$y > 0$	$x = 0$	0
	else	ModSwitch$(+, x, y)$
$y < 0$		ModSwitch$(-, -x, -y)$

#1#2: y
#3#4: x

```
\def\BIC@ModSwitchSign#1#2!#3#4!{%
```

```
  \ifcase\ifx\\#2\\%
           \ifx#100 % y = 0
           \else1 % y > 0
           \fi
          \else
            \ifx#1-2 % y < 0
            \else1 % y > 0
            \fi
          \fi
    \BIC@AfterFi{ 0\BigIntCalcError:DivisionByZero}%
  \or % y > 0
    \ifcase\ifx\\#4\\\ifx#300 \else1 \fi\else1 \fi % x = 0
      \BIC@AfterFiFi{ 0}%
    \else
      \BIC@AfterFiFi{%
        \BIC@ModSwitch{}#3#4!#1#2!%
      }%
    \fi
  \else % y < 0
    \ifcase\ifx\\#4\\%
             \ifx#300 % x = 0
             \else1 % x > 0
             \fi
           \else
             \ifx#3-2 % x < 0
             \else1 % x > 0
             \fi
           \fi
      \BIC@AfterFiFi{ 0}%
    \or % x > 0
      \BIC@AfterFiFi{%
        \BIC@ModSwitch--#3#4!#2!%
      }%
    \else % x < 0
      \BIC@AfterFiFi{%
        \BIC@ModSwitch-#4!#2!%
      }%
    \fi
  \BIC@Fi
}
```

`\BIC@ModSwitch` Decision table for `\BIC@ModSwitch`.

$y = 1$		0
$y = 2$	ifodd(x)	sign 1
	else	0
$y > 2$	$x < 0$	$z \leftarrow x - (x/y) * y; \quad (z < 0)\,?\,z + y : z$
	$x > 0$	$x - (x/y) * y$

#1: sign
#2#3: x
#4#5: y

```
\def\BIC@ModSwitch#1#2#3!#4#5!{%
  \ifcase\ifx\\#5\\%
           \ifx#410 % y = 1
           \else\ifx#421 % y = 2
           \else2 % y > 2
           \fi\fi
```

```
          \else2 % y > 2
          \fi
    \BIC@AfterFi{ 0}% y = 1
  \or % y = 2
    \ifcase\BIC@ModTwo#2#3! % even(x)
      \BIC@AfterFiFi{ 0}%
    \or % odd(x)
      \BIC@AfterFiFi{ #11}%
? \else\BigIntCalcError:ThisCannotHappen%
    \fi
  \or % y > 2
    \ifx\\#1\\%
    \else
      \expandafter\BIC@Space\romannumeral0%
      \expandafter\BIC@ModMinus\romannumeral0%
    \fi
    \ifx#2-% x < 0
      \BIC@AfterFiFi{%
        \expandafter\expandafter\expandafter\BIC@ModX
        \bigintcalcSub{#2#3}{%
          \bigintcalcMul{#4#5}{\bigintcalcDiv{#2#3}{#4#5}}%
        }!#4#5!%
      }%
    \else % x > 0
      \BIC@AfterFiFi{%
        \expandafter\expandafter\expandafter\BIC@Space
        \bigintcalcSub{#2#3}{%
          \bigintcalcMul{#4#5}{\bigintcalcDiv{#2#3}{#4#5}}%
        }%
      }%
    \fi
? \else\BigIntCalcError:ThisCannotHappen%
  \BIC@Fi
}
```

\BIC@ModMinus

```
\def\BIC@ModMinus#1{%
  \ifx#10%
    \BIC@AfterFi{ 0}%
  \else
    \BIC@AfterFi{ -#1}%
  \BIC@Fi
}
```

\BIC@ModX #1#2: z
#3: x

```
\def\BIC@ModX#1#2!#3!{%
  \ifx#1-% z < 0
    \BIC@AfterFi{%
      \expandafter\BIC@Space\romannumeral0%
      \BIC@SubXY#3!#2!!!%
    }%
  \else % z >= 0
    \BIC@AfterFi{ #1#2}%
  \BIC@Fi
}
```

```
\BIC@AtEnd%
```

⟨/package⟩

3 Installation

3.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/bigintcalc/bigintcalc.dtx The source file.

CTAN:macros/latex/contrib/bigintcalc/bigintcalc.pdf Documentation.

Bundle. All the packages of the bundle 'bigintcalc' are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN:install/macros/latex/contrib/bigintcalc.tds.zip

TDS refers to the standard "A Directory Structure for TEX Files" (CTAN:pkg/tds). Directories with `texmf` in their name are usually organized this way.

3.2 Bundle installation

Unpacking. Unpack the `bigintcalc.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip bigintcalc.tds.zip -d ~/texmf
```

3.3 Package installation

Unpacking. The `.dtx` file is a self-extracting docstrip archive. The files are extracted by running the `.dtx` through plain TEX:

```
tex bigintcalc.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```
bigintcalc.sty → tex/generic/bigintcalc/bigintcalc.sty
bigintcalc.pdf → doc/latex/bigintcalc/bigintcalc.pdf
bigintcalc.dtx → source/latex/bigintcalc/bigintcalc.dtx
```

If you have a `docstrip.cfg` that configures and enables docstrip's TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

3.4 Refresh file name databases

If your TEX distribution (TEX Live, MiKTEX, ...) relies on file name databases, you must refresh these. For example, TEX Live users run `texhash` or `mktexlsr`.

3.5 Some details for the interested

Unpacking with LATEX. The `.dtx` chooses its action depending on the format:

plain TEX: Run docstrip and extract the files.

LATEX: Generate the documentation.

[1] CTAN:pkg/bigintcalc

If you insist on using LaTeX for docstrip (really, docstrip does not need LaTeX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{bigintcalc.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfLaTeX:

```
pdflatex bigintcalc.dtx
makeindex -s gind.ist bigintcalc.idx
pdflatex bigintcalc.dtx
makeindex -s gind.ist bigintcalc.idx
pdflatex bigintcalc.dtx
```

4 History

[2007/09/27 v1.0]

- First version.

[2007/11/11 v1.1]

- Use of package pdftexcmds for LuaTeX support.

[2011/01/30 v1.2]

- Already loaded package files are not input in plain TeX.

[2012/04/08 v1.3]

- Fix: pdftexcmds wasn't loaded in case of LaTeX.

[2016/05/16 v1.4]

- Documentation updates.

[2019/12/15 v1.5]

- Documentation updates.

5 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

A

B

C

E

I

N

P

R

T

W

X