Main, BSA GetOperator

BSA GetSignedInt //

STA Num

ISZ FlagLeftOperand //Left operand was assigned

BSA GetSignedInt

STA Num2

BUN FI

//Main Data

Num, DEC 0

Num2, DEC 0

Res, DEC 0

//

GetOperator, HEX 0 // HERE IT ALL STARTS

CLA

BSA In\_char

CheckMul, CMA // IF One of the operators == zero then

INC

ADD Mul

SZA

BUN CheckPlus

STA Mul

BUN GetOperator I

CheckPlus, LDA TOperator

CMA

INC

ADD Plus

SZA

BUN CheckMinus

STA Plus

BUN GetOperator I

CheckMinus, LDA TOperator

CMA

INC

ADD Minus

SZA

BUN CheckDiv

STA Minus

BUN GetOperator I

CheckDiv, LDA TOperator

CMA

INC

ADD Div

SZA

BUN FI

STA Div

BUN GetOperator I

GetSignedInt, HEX 0

CLA

STA TNum

LDA FlagOFF

STA minus\_flagRight

LDA FlagLeftOperand

SZA

BUN In\_char

ISZ minus\_flagLeft

In\_char, HEX 0

BSA Getc

STA Cc

LDA operatorFlag

SZA

BUN MinusMinus //IF OPERATOR START

LDA Cc

STA TOperator

LDA operatorFlag

INC

STA operatorFlag

LDA TOperator

BUN In\_char I

MinusMinus, LDA Cc

ADD minusMinusASCII

SZA

BUN operatorFlagCheck

LDA FlagOn //FlagOn starts with negative number

STA minus\_flagRight

BSA Getc

STA Cc

operatorFlagCheck, LDA OperatorFlag

SZA

BUN getUnsigned //if operator phase is already executed

BUN GetSignedInt I

getUnsigned, LDA Cc

ADD minusCReturn

SZA

BUN Convert

BUN End\_loop

Convert, LDA Cc // IF Character is not ENTER or Operator

ADD mASCII\_zero //CONVERT TO the real number - 30

STA Cc

DigitCase, LDA TNum // The current whole number we have

BSA MultBy10

ADD Cc // Cc is currently the original next number

STA TNum // TNum = (prev digit\*10) + currentDigit

BUN In\_char

End\_loop, LDA minus\_flagRight

SPA

BUN FI\_1

LDA TNum

CMA

INC

STA TNum

FI\_1, LDA TNum

BUN GetSignedInt I

FI, HEX 0

SENDTOPLUS, LDA Plus

SZA

BUN SENDTOMINUS

LDA Num

ADD Num2

STA Res

BUN PRINTRES

SENDTOMINUS, LDA Minus

SZA

BUN SENDTOMUL

LDA Num2

CMA

INC

ADD Num

STA Res

BUN PRINTRES

SENDTOMUL, LDA Mul

SZA //if(mul == 0)

BUN SENDTODIV

//THEN MULTIPLY

PreMul, LDA minus\_flagRight

ADD minus\_flagLeft

STA resMinusFlag //IF(resMinusFlag == 1) THEN res = ‘-num’

CMA

INC

ADD Minus1

Multiply,

LDA DigitCount // FOR (each 16 binary digits in multiplier)

ADD Digits //

SZA //

BUN BodyMul //

BUN PRINTRES //

//

BodyMul, LDA Num2 // DO

CIR //

STA Num2 //

SZE // IF (digit == 1)

BUN THENMUL //

BUN FI\_MUL //

THENMUL, LDA Res // THEN result = result + multiplicand;

ADD Num //

STA Res //

FI\_MUL, LDA Num // FI;

CIL // Shift(multiplicand) 1 place to left;

STA Num //

ISZ DigitCount // DigitCount++;

BUN Multiply // OD;

BUN PRINTRES

SENDTODIV, HLT

//

// main() data

DigitCount, DEC 0

Digits, DEC -16

PRINTRES, LDA Res

BSA putSignedIntt // PutUnsignedIntt(Num);

HLT //

NumMinus, DEC -32768 //

putSignedIntt, HEX 0 // putSignedIntt(short signed int Tnum) {

STA Tnum //

LDA Power10Ptr //

STA Ptr // Ptr = @Power10Array

LDA It\_count

STA Count // Count = It\_count; i.e. 4 in the case of 16 bits

LDA TNum // IF (TNum 0; Count--)

SNA //

BUN Positive

Negative, LDA TNum

CMA

INC

STA Stripped

LDA ProtoMinus

BSA Putc

BUN ForLoop

Positive, LDA TNum

STA Stripped

ForLoop, LDA Count

SPA

BUN EndFor //

CLA // DO digit =0;

STA Digit //

Loop, LDA ptr I // WHILE (Stripped - \*Ptr > 0)

ADD Stripped // DO

SNA

BUN Continue //

BUN Outx //

Continue, STA Stripped // Stripped=Stripped - \*Ptr;

ISZ digit // digit++;

BUN Loop // OD;

Outx, LDA digit //

ADD ascii\_Offset //

BSA Putc //

ISZ ptr // ptr++;

LDA count // Count--;

ADD Minus1 //

STA Count //

BUN ForLoop // OD;

EndFor, LDA Stripped //

ADD ascii\_Offset //

BSA Putc // Output(last digit);

// print units – the left over in ACC

End, BUN putSignedIntt I // return; }

// putSignedIntt data

digit, DEC 0

ascii\_Offset, HEX 30 // digit to ascii representation offset

Stripped, DEC 0 // +ve value of TNum

Minus1, DEC -1

count, DEC 0

It\_count , DEC 4 // loop count (for 16 bit integer)

Power10Ptr, HEX 100 // @Power10Array

ptr, HEX 0 // //

// GetSignedInt Data

minusMinusASCII, HEX -2D

OperatorFlag, DEC 0

minus\_flagRight, DEC 0 //If is 1 then rightOperand is Negative

minus\_flagLeft, DEC 0

FlagOn, DEC 1

FlagOff, DEC 0

FlagLeftOperand, DEC 0 //If left operand was assigned

// GetOperatorData

TOperator, DEC 0

Mul, HEX 2A

Minus, HEX 2D

Plus, HEX 2B

Div, HEX 2F

ProtoMul, HEX 2A // Prototype will reassign \*/+- to the original.

ProtoMinus, HEX 2D

ProtoPlus, HEX 2B

ProtoDiv, HEX 2F

//GetUnsignedInt Data

TNum, DEC 0

Cc, DEC 0

mASCII\_zero, HEX -30

minusCReturn, HEX -D

MultBy10, HEX 0

CLE

CIL

STA tmp

CIL

CIL

ADD tmp

BUN MultBy10 I

// MultBy10 data

tmp, DEC 0

// subroutine getC()

Getc, HEX 0

Inp\_char, SKI

BUN Inp\_char

INP

Echo, SKO

BUN Echo

OUT

BUN Getc I

ORG 100

Power10Array, DEC -10000 // -10 to power of 4

DEC -1000 // -10 to power of 3

DEC -100 // -10 to power of 2

DEC -10 // -10 to power of 1

//Subroutine to print a char to screen

Putc, HEX 0 // void Putc(char) {

Out, SKO

BUN Out

OUT // print(char);

BUN Putc I // }

END