//============OPENING SENTENCE

OPENING, LDA StrPTROP I

SZA

BUN PRINTO

BUN Main

PRINTO, LDA STRPTROP I

OUT

ISZ STRPTROP

BUN OPENING

Main, BSA GetOperator

STA StrPTRTEMP

BSA GetSignedInt //

STA Num

ISZ FlagLeftOperand //Left operand was assigned

BSA GetSignedInt

STA Num2

BUN FI

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 OperatorFlag

SZA //If(operatorFlag == 0)

BUN checkLeftOperand

ISZ OperatorFlag //THEN

BUN In\_char

checkLeftOperand, LDA FlagLeftOperand

SZA

BUN In\_char

ISZ minus\_flagLeft

In\_char, HEX 0

BSA Getc

STA Cc

checkX, ADD minusX

SZA //IF minusX == 0

BUN ContinueCheck

HLT //THEN

ContinueCheck, 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

PreMul, LDA minus\_flagRight //THEN

ADD minus\_flagLeft

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

////////////TRIAL

gotoLeftOperand, LDA minus\_flagLeft

ADD Minus1

SZA //If(minus\_flagLeft) == 1

BUN gotoRightOperand

LDA Num //THEN

CMA

INC

STA Num

gotoRightOperand, LDA minus\_flagRight

ADD Minus1

SZA

BUN Multiply

LDA Num2

CMA

INC

STA Num2

///////////////////////////////

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, LDA Num2

SZA //IF(rightOperand == 0)

BSA DIV\_FUNC //DIV\_FUNC();

HLT

PRINTDIVZERO, LDA STRPTRDIVZERO I //THEN

SZA //IF StrPtrDivZero == 0

BUN CONTINUESTRDIV0

HLT //

CONTINUESTRDIV0, OUT //THEN

ISZ STRPTRDIVZERO

BUN PRINTDIVZERO

PleaseEnterNum, HEX 0

startPrinting, LDA StrPTRTEMP I

SZA //if (\*strptrtemp == 0) {

BUN PrintOutLetter

BUN PleaseEnterNum I

PrintOutLetter, OUT //THEN

ISZ StrPTRTEMP

BUN startPrinting

DIV\_FUNC, HEX 0

LDA Num //DIV\_FUNC() {

chckMinusFlagR, BSA PosOrNeg

STA minus\_flagLeft

checkMinusFlagL, LDA Num2

BSA PosOrNeg

STA minus\_flagRight

ADD minus\_flagLeft

STA resMinusFlag

LeftToPositive, LDA minus\_flagLeft

SPA //if(minus\_flagLeft == 1)

BUN RightToPositive

LDA Num

CMA //THEN

INC

STA Num

RightToPositive, LDA minus\_flagRight

SPA //if(minus\_flagRight == 1)

BUN positiveResultDiv

LDA Num2 //THEN

CMA

INC

STA Num2

positiveResultDiv, LDA Num2 //two operands are positive

CMA

INC

ADD Num

SPA //if(Num > Num2)

BUN DIV\_FUNC I

FOR\_LOOPDIV1, LDA Num2 //THEN

CMA

INC

ADD Num

STA Num

SNA //if(acc < 0)

BUN counterPlusDiv

FIDIV, ADD Num2 //THEN

STA remainder

BUN DIV\_FUNC I

counterPlusDiv, ISZ counterDIV

BUN FOR\_LOOPDIV1

//}

//This function returns to tmpMSB true (negative number) or false (positive number)

PosOrNeg, HEX 0

CLE

CIL

CLA

CIL

BUN PosOrNeg I

// main() data

remainder, DEC 0

tmpNum, DEC 0

counterDIV, DEC 0

tmpMSB, DEC 0

StrPTRTEMP, HEX 0

DigitCount, DEC 0

Digits, DEC -16

resMinusFlag, DEC 0

STRPTRDIVZERO, HEX 500

StrPTROP, HEX 300 //

Num, DEC 0

Num2, DEC 0

Res, DEC 0

NumMinus, DEC -32768 //

StrDivBy0, HEX 500

PRINTRES, LDA Mul

SZA //IF ITS MUL

BUN STEP2

BSA convertResultOfMul //THEN

STEP2, CLE

LDA Equal

OUT

LDA Res

BSA putSignedIntt // PutUnsignedIntt(Num);

HLT //

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

SZA //if digit == 0

BUN printDigit

checkFlagZero, LDA flagZero // THEN

SZA //If flagZero == 0 && digit==0

addIT, ISZ flagZero

addIT2, BUN gotoNextDigit //THEN

printDigit, ADD ascii\_Offset //

BSA Putc //

gotoNextDigit, 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; }

ConvertResultOfMul, HEX 0

LDA resMinusFlag

ADD Minus1

SZA //If(resMinusFlag == 1)

BUN ConvertResultOfMul I

LDA Res //THEN

CMA

INC

STA Res

BUN ConvertResultOfMul I

// 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 400 // @Power10Array

ptr, HEX 0 // //

// GetSignedInt Data

minusMinusASCII, HEX -2D

OperatorFlag, DEC 0

minus\_flagLeft, DEC 0

minus\_flagRight, DEC 0

flagZero, DEC 0

FlagOn, DEC 1

FlagOff, DEC 0

FlagLeftOperand, DEC 0 //If left operand was assigned

// GetOperatorData

TOperator, DEC 0

Mul, HEX 2A

//ALL OPERATORS

minusX, HEX -58

Minus, HEX 2D

Plus, HEX 2B

Div, HEX 2F

Equal, HEX 3D

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

//==================DATA OF OPENING SENTENCE

ORG 300

Str, HEX 4F // 'Op available: +-\*/. Input is terminated in a single

// line after =. To exit program type X.'

HEX 70

HEX 20

HEX 61

HEX 76

HEX 61

HEX 69

HEX 6C

HEX 61

HEX 62

HEX 6C

HEX 65

HEX 3A

HEX 20

HEX 2B

HEX 20

HEX 2D

HEX 20

HEX 2A

HEX 20

HEX 2F

HEX 2E

HEX 20

HEX 50

HEX 72

HEX 65

HEX 73

HEX 73

HEX 20

HEX 58

HEX 20

HEX 74

HEX 6F

HEX 20

HEX 65

HEX 78

HEX 69

HEX 74

HEX D

HEX 4F

HEX 70

HEX 3A

DEC 0 // null - end of string = '\0'

ORG 400

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 // }

//DATA OF ERROR MSG DIVISION BY ZERO!

ORG 500

errorDivZero, HEX 45

HEX 52

HEX 52

HEX 4F

HEX 52

HEX 3A

HEX 20

HEX 44

HEX 49

HEX 56

HEX 49

HEX 53

HEX 49

HEX 4F

HEX 4E

HEX 20

HEX 42

HEX 59

HEX 20

HEX 30

DEC 0 // null - end of string = '\0'

END