//Everything that is highlighted with grey is the main program I divided the functions correspondingly

OPENING, LDA StrPTROP //THEN

BSA PrintString

BUN Main

NotFirstTime, LDA StrOpInput

BSA PrintString

Main, BSA GetOperator // TOperator = GetOperator();

LDA StrPTRNum1

STA strTemp

BSA PrintString

BSA GetSignedInt // Num = GetSignedInt();

STA Num

ISZ FlagLeftOperand //Left operand indication was assigned

LDA StrPTRNum2

STA strTemp

BSA PrintString

BSA GetSignedInt // Num2 = GetSignedInt();

STA Num2

BUN FI //result = FI();

//GetOperator() is a function that will wait for an operator input from the user and determine the input //accordingly and will change the right operator variable to zero so the main will know which operator // to continue calculating with.

GetOperator, HEX 0 //

CLA

BSA In\_char

CheckMul, CMA

INC

ADD Mul

SZA

BUN CheckPlus

isMul, STA Mul

BUN GetOperator I

CheckPlus, LDA TOperator

CMA

INC

ADD Plus

SZA //if(TOperator == plus)

BUN CheckMinus

isPlus, STA Plus //THEN

BUN GetOperator I

CheckMinus, LDA TOperator

CMA

INC

ADD Minus

SZA

BUN CheckDiv

isMinus, STA Minus

BUN GetOperator I

CheckDiv, LDA TOperator

CMA

INC

ADD Div

SZA

BUN errIsNotOp

isDiv, STA Div

BUN GetOperator I

errIsNotOp, LDA cReturn

OUT

LDA Cc

OUT

LDA StrNotOp

BSA PrintString

BUN Reset

//This function will get any number and store it in variable ‘Num’

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

SPA //if(FlagLeftOperand > 0)

BUN In\_char

ISZ FlagLeftOperand //THEN

//this function determines between different types of user input to know how to handle the input num

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

limit0To9, LDA Cc //Cc is now an unsigned decimal digit

SNA //if(digit < 0)

BUN ContinueLimit0to9

BUN errIsNotDigit

ContinueLimit0to9, LDA Cc //THEN

ADD minus\_nine

SPA //if(digit < 0 OR digit - 9 > 0)

BUN DigitCase //ELSE digit is fine

BUN errIsNotDigit //THEN digit Cc is NOT ok (between 0 - 9)

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

//THE FI FUNCTION IS THE CALCULATION OF THE RESULT

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’

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;

SENDTODIV, LDA Num2

SZA //IF(rightOperand == 0)

BUN ELSE1 //DIV\_FUNC();

BUN PRINTDIVZERO //THEN

ELSE1, BSA DIV\_FUNC

PrintResDiv, LDA counterDIV //THEN

STA Res

LDA resMinusFlag

ADD Minus1

SZA //if(resMinusFlag == 1)

BUN OutPutResDiv

LDA Res //THEN

CMA

INC

STA Res

OutPutResDiv, LDA Res

BSA putSignedIntt

LDA openParenthesis

OUT

CLA

STA flagZero

LDA remainder

STA Res

BSA putSignedIntt

LDA closedParenthesis

OUT

BUN Reset

PRINTDIVZERO, LDA STRPTRDIVZERO I //THEN

SZA //IF StrPtrDivZero == 0

BUN CONTINUESTRDIV0

BUN Reset //

CONTINUESTRDIV0, OUT //THEN

ISZ STRPTRDIVZERO

BUN PRINTDIVZERO

//DIV\_FUNC In this function I am converting first all the operands to positive and later make the //operation between them and then turn the result to negative or positive accordingly (if Negative //AND Negative OR Negative AND Positive OR Positive AND Negative OR Positive AND Positive) //according to the different operators

DIV\_FUNC, HEX 0 //DIV\_FUNC() {

LDA Num

chckMinusFlagR, BSA PosOrNeg

STA minus\_flagLeft

checkMinusFlagL, LDA Num2

BSA PosOrNeg

STA minus\_flagRight

ADD minus\_flagLeft

STA resMinusFlag

//Checking that resMinusFlag is equal one than the result will be negative number

equalOne, LDA Num

CMA

INC

ADD Num2

SZA //if(Num == Num2 or -num == -num2)

BUN checkOpose

LDA One //THEN

STA counterDiv

LDA zero

STA remainder

BUN DIV\_FUNC I

checkOpose, LDA Num

ADD Num2

SZA //if(num == -num2 or -num == num2)

BUN LeftToPositive

LDA One //THEN

STA counterDiv

LDA zero

STA remainder

BUN DIV\_FUNC I

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 remainderLeft

//In this phase we turned all of our operands to positive so we can do the division operation (by doing subtraction) and at the end we will return the result to positive or negative (if resMinusFlag is 0 or two than its positive if its 1 than its negative)

//WHILE(Num >= 0) {

// Num -= Num2

// counterDiv++; // counterDiv is the quotient

//}

//remainder = Num + Num2; //After the process num+num2 = remainder

//

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

remainderLeft, LDA minus\_flagLeft

SZA //if(minus\_flagLeft == 0)

BUN convertRemainder

LDA Num // THEN

STA remainder

BUN DIV\_FUNC I

convertRemainder, LDA Num

CMA

INC

STA remainder

BUN DIV\_FUNC I

errIsNotDigit, LDA cReturn

OUT

LDA Cc

ADD ascii\_Offset

OUT

LDA StrNotDigit

STA strTemp

BSA PrintString

BUN Reset

//The PosOrNeg function returns true (negative number) or false (positive number)

PosOrNeg, HEX 0

CLE

CIL

CLA

CIL

BUN PosOrNeg I

// main() data

strTemp, HEX 0

minus\_nine, DEC -9

openParenthesis, HEX 28

closedParenthesis, HEX 29

remainder, DEC 0

One, DEC 1

tmpNum, DEC 0

counterDIV, DEC 0

StrPTRTEMP, HEX 0

four, DEC 4

DigitCount, DEC 0

Digits, DEC -16

resMinusFlag, DEC 0

STRPTRDIVZERO, HEX 600

StrPTROP, HEX 400 //

StrNotOp, HEX 430

StrOpInput, HEX 325

StrNotDigit, HEX 450

StrPTRNum1, HEX 470

StrPTRNum2, HEX 480

copyStrOpInput, HEX 325

copySTRPTRDIVZERO, HEX 600

copyStrPTROP, HEX 400 //

copyStrNotDigit, HEX 450

copyStrPTRNum1, HEX 470

copyStrPTRNum2, HEX 480

Num, DEC 0

Num2, DEC 1

Res, DEC 0

NumMinus, DEC -32768 //

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 500 // @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

//ALL OPERATORS

minusX, HEX -58

Mul, HEX 2A

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

zero, DEC 0

//GetUnsignedInt Data

TNum, DEC 0

Cc, DEC 0

mASCII\_zero, HEX -30

minusCReturn, HEX -D

cReturn, HEX D

PRINTRES, LDA Mul

SZA //IF ITS MUL

BUN STEP2

BSA convertResultOfMul //THEN

STEP2, CLE

LDA Equal

OUT

LDA Res

BSA putSignedIntt // PutUnsignedIntt(Num);

BUN Reset //

//putSignedIntt function will print all the digits of the result to the screen one by one

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

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)

SNA //

BUN Positive

Negative, LDA TNum //THEN

CMA //Stripped = -TNum;

INC

STA Stripped

LDA ProtoMinus

BSA Putc // putc(“-”);

BUN ForLoop

Positive, LDA TNum // ELSE Stripped = TNum;

STA Stripped // FI

ForLoop, LDA Count //FOR(Count=It\_count;Count>0;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 //else

checkFlagZero, LDA flagZero //THEN

SZA //if(flagZero==0)

prePrintDigit, BUN COUNTERIT //else flagZero++

BUN gotoNextDigit //then gotoNextDigit

COUNTERIT, ISZ flagZero

printDigit, ISZ flagZero

LDA digit

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

///PrintString will print any string in address of strTemp

PrintString, HEX 0 //PrintString(strTemp);

STA strTemp

forLoopString, LDA strTemp I

SZA //if(\*strTemp == 0)

BUN PrePUTC

BUN PrintString I //THEN

prePUTC, BSA Putc

ISZ strTemp

BUN forLoopString

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

//After the calculation we must reinitialize all our variables

Reset, LDA ProtoMinus

STA Minus

LDA ProtoPlus

STA Plus

LDA ProtoMul

STA Mul

LDA ProtoDiv

STA Div

LDA zero

STA Num

STA resMinusFlag

STA Num2

STA counterDIV

STA count

STA digit

STA strTemp

STA tmp

STA tmpNum

STA DigitCount

STA operatorFlag

STA FlagLeftOperand

STA flagZero

STA Stripped

STA minus\_flagRight

STA minus\_flagLeft

STA OperatorFlag

STA flagZero

STA ptr

STA Res

STA TNum

STA Cc

LDA four

STA It\_count

LDA copyStrOpInput

STA StrOpInput

LDA copySTRPTRDIVZERO

STA STRPTRDIVZERO

LDA copyStrNotDigit

STA StrNotDigit

LDA copyStrPTRNum1

STA StrPTRNum1

LDA copyStrPTRNum2

STA StrPTRNum2

LDA copyStrPTROP

STA StrPTROP

CLA

CLE

BUN NotFirstTime

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

ORG 400

Str, HEX 0D

HEX 4F

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 325

OpInput, HEX D

HEX 4F

HEX 70

HEX 3A

DEC 0

ORG 430

CcNotOp, HEX 20

HEX 69

HEX 73

HEX 20

HEX 6E

HEX 6F

HEX 74

HEX 20

HEX 61

HEX 6E

HEX 20

HEX 6F

HEX 70

HEX 65

HEX 72

HEX 61

HEX 74

HEX 6F

HEX 72

DEC 0

ORG 450

CcNotDigitStr, HEX 20

HEX 69

HEX 73

HEX 20

HEX 6E

HEX 6F

HEX 74

HEX 20

HEX 61

HEX 20

HEX 64

HEX 69

HEX 67

HEX 69

HEX 74

DEC 0

ORG 470

EnterNum1STR, HEX 0D

HEX 4E

HEX 75

HEX 6D

HEX 31

HEX 3A

DEC 0

ORG 480

EnterNum2STR, HEX 0D

HEX 4E

HEX 75

HEX 6D

HEX 32

HEX 3A

DEC 0

ORG 500

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 600

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