**Basic LC-3 Calculator**

Utah Valley University

CS 2810-002

Computer Organization and Architecture

By

Russell Andlauer

Ernie Vilela

Michael De La Barra

**Table of Contents**

1. **Abstract**
2. **Introduction**
3. **Technical Sections**
4. **Discussion of the work**
5. **Conclusions**
6. **References**

**1. Abstract**

With this research project, we aim to successfully build a Basic Calculator by using the LC-3 Assembly Language. This calculator will take two individual integers in separate instances as inputs. This calculator will have five different conditional branches that will perform basic arithmetic operations, such as add, subtract, and multiply. It will display the solutions to those four different operations. At the completion of the different arithmetic operations, the calculator will display those results. The main incentive that sparked our desire to work on this project was to assemble a useful tool that will allow us to apply the knowledge gained in this class, Computer Organization and Architecture. The main design of our project is divided in several parts of that make use of the Assembly Language. The first part is the main code of the project, the following part is where the subroutines are declared for each of the five operations that the calculator has to perform, and the last part contains the static memory location. For the completion of this project the code was written in small parts, and test it periodically, so that all the code could work correctly.

**2. Introduction**

Calculators were created to reduce the amount of time and mental effort in arithmetical calculations, and to reduce or eliminate chance of a people’s making mistakes when doing calculations. There are two types of calculators; mechanical calculator and software calculator. The majority of the mechanical calculators were as big in size to small desktop computers. Now they have been considered obsolete after the creation of the software calculator. Computers have some complex usages that are not limited to simple calculation, but back in the early years computers were built to perform calculations.

A software calculator is a type of calculator that has been implemented as programs which are developed in computers to make our lives easier. A basic software calculator is able to set operation for the user to be selected one at the time. It also can be used to perform any process that implicates a sequence of steps, each of which applies the operation selected. A calculator has a small group of functions that allows us to execute simple operations. A simple software calculator would not be able to accept large amounts of input data from the user or produce multiple results.

The LC-3 assembly language is a rich, and detailed programming language that allow us to interact with part of the memory, as well as with the user by means of inputs and outputs. The use of the assembly language makes the programming process more user-friendly than programming in machine language. The assembly language is comprised of various parts, which make the structure and organization easier to manipulate, and to work with. One of the tools that is featured in this language are the instructions that allow us to interact directly with the memory. These instructions grant us the possibility to handle inputs from the user, and to perform several operations, such as addition, loading, or output of numbers and letters. Another important tool that is featured in this language is the assembler directives (or also called pseudo-ops.) These pseudo-ops provide the user with the ability to tell the assembler where in memory to place the LC-3 program(.ORIG), to tell the assembler to set aside the next location in the program and initialize with a value of the operand(.FILL), to tell the assembler to set aside some number sequential memory locations in the program(.BLKW), to tell the assembler to initialize a sequence of n+1 memory locations(.STRINGZ), and to tell the assembler where the program ends (.END). These basic tools from the LC-3 Assembly Language are going to be utilized in order to complete this project.

3. **Technical Sections**

The emphasis of this project is to be able to use the resources offered by the assembly language to create a basic calculator that performs the operations of addition, subtraction, and multiplication. In order to achieve this, an analytical revision of each of the arithmetic operators mentioned had to be made by our group.

**Addition**

The addition of two numbers is simply taking the value of a first number, and annex this number with a second number. Therefore, producing a third number with a greater value than both number imputed.

Example:

4 + 3 = 7

**Subtraction**

The subtraction of two numbers is taking the value of a first number, and deducted from a second number. In other words, taking out certain quantity from a certain number.

Example:

4 - 3 = 1

**Multiplication**

The multiplication of two numbers can be found by taking the value of a first numbers, and add the same numbers as many times as it is indicated by the second number.

Example:

4 \* 3 = 4 + 4 + 4 = 12

4 is added 3 times

**4. Discussion of the work**

This is the complete code of our calculator:

; Program: Calculator\_Group\_Project.asm

; Project#: Calculator Group Project

; Name: Russell Andlauer, Michael De La Barra, Ernie Vilela

; Class: CS-2810-002

; Date: 10 Dec 2014

; Version: 1.0

;

; Ethics: I declare that the following source code was written

; solely by me. I understand that copying any source

; code, in whole or in part, constitutes cheating,

; and that I will receive a zero grade on this

; project if I am found in violation of this ethic.

;

; X Russell Andlauer, Michael De La Barra, Ernie Vilela

;

; Description:

; A calculator that can add, subtract, multiply, and clear two numbers or negate a number

;

;-------1---------2---------3---------4---------5---------6---------7---------

.ORIG x3000 ;start of program

; Class Header

LEA R0, classHeader ;point to class string

trap x22 ;print string out

LD R0, newLine ;get carriage return

OUT ;print it out

;

; Start of main program

;

; Ask the user to input 1st number between 0-9 (echo)

;

LEA R0, promptNum

PUTS

GETC

OUT

;

; Convert the entered number from ASCII to binary and move it into R1

;

JSR ASCIItoBinary

AND R1, R1, #0

ADD R1, R0, R1

;

; Ask the user to input the 2nd number between 0-9 (echo)

LEA R0, promptNum

PUTS

GETC

OUT

; Convert the entered number from ASCII to binary and move it into R2

;

JSR ASCIItoBinary

AND R2, R2, #0

ADD R2, R0, R2

;

; Ask the user to input the operation to be performed (echo) (Exit, +, -, \*, NOT, AND)

;

LEA R0, promptOperator

PUTS

LEA R0, opKey

PUTS

LEA R0, promptAnswer

PUTS

LEA R0, promptExit

PUTS

GETC

OUT

; Perform the operation and save the resulting number in R3 (echo)

;

LD R4, Plus

ADD R4, R4, R0

BRz GoOpAdd

;

LD R4, Minus

ADD R4, R4, R0

BRz GoOpSubtract

;

LD R4, Multi

ADD R4, R4, R0

BRz GoOpMulti

;

LD R4, N

ADD R4, R4, R0

BRz GoOpNegate

;

LD R4, C

ADD R4, R4, R0

BRz GoOpClear

;

BRnp Exit

;

GoOpAdd JSR OpAdd

OUT

BRnzp Exit

;

GoOpSubtract JSR OpSubtract

OUT

BRnzp Exit

;

GoOpMulti JSR OpMulti

OUT

BRnzp Exit

;

GoOpNegate JSR OpNegate

LD R4, ASCIINegSign

AND R5, R5, #0

ADD R5, R5, R0

AND R0, R0, #0

ADD R0, R0, R4

OUT

AND R0, R0, #0

ADD R0, R5, R0

OUT

BRnzp Exit

;

GoOpClear JSR OpClear

BRnzp Exit

;

Exit AND R0, R0, #0

;

LEA R0, termination ;point to termination string

PUTS

trap x25 ;stop the program

;

;=====================================================

;Subroutines

;

ASCIItoBinary AND R4, R4, #0

LD R4, NegASCIIOffset

ADD R0, R0, R4

RET

;

OpAdd AND R3, R3, #0

ADD R3, R1, R2

LD R4, PosASCIIOffset

ADD R0, R3, R4

RET

;

OpSubtract AND R3, R3, #0

NOT R2, R2

ADD R2, R2, #1

LD R4, PosASCIIOffset

AND R0, R0, #0

ADD R3, R1, R2

ADD R0, R3, R4

;ADD R0, R0, R5

RET

;

OpMulti AND R3, R3, #0

AND R5, R5, #0

ADD R5, R1, R5

;

MultiLoop;

ADD R3, R3, R2

ADD R5, R5, #-1

BRp MultiLoop

;

LD R4, PosASCIIOffset

AND R0, R0, #0

ADD R0, R4, R3

RET

;

OpNegate AND R3, R3, #0

ADD R3, R2, R3

ADD R5, R2, R2; Save double input value in R5 for later ASCII conversion

NOT R3, R3

ADD R3, R3, #1

LD R4, PosASCIIOffset

AND R0, R0, #0

ADD R0, R3, R4

ADD R0, R0, R5

RET

;

OpClear AND R0, R0, #0

AND R1, R1, #0

AND R2, R2, #0

AND R3, R3, #0

AND R4, R4, #0

AND R5, R5, #0

RET

;-------1---------2---------3---------4---------5---------6---------7---------

;

; Static Memory Allocation

;

;-------1---------2---------3---------4---------5---------6---------7---------

answer .BLKW x1 ;save for answer

;constants

classHeader .STRINGZ "CS 2810\nCalculator Project\n\n"

newLine .FILL x0A ;newLine character

promptNum .STRINGZ "\nEnter a number 0-9\n"

promptAnswer .STRINGZ "\nAnswer range valid from 0-9"

promptOperator .STRINGZ "\nChoose operation: +, -, \*, n, or c\n"

opKey .STRINGZ "\nn = negate, c = clear\n"

promptExit .STRINGZ "\nEnter any key to Exit\n"

NegASCIIOffset .FILL xFFD0

PosASCIIOffset .FILL x0030

ASCIINegSign .FILL x002D

Plus .FILL xFFD5

Minus .FILL xFFD3

Multi .FILL xFFD6

N .FILL xFF92

C .FILL xFF9D

termination .STRINGZ "\nExiting"

; ///////////////////////////////////////////////////////////////////////////////////////////////

; Attempted to create a lookup table to use for displaying double digit answers

;LookUp10 .FILL #0

;.FILL #10

;.FILL #20

;.FILL #30

;.FILL #40

;.FILL #50

;.FILL #60

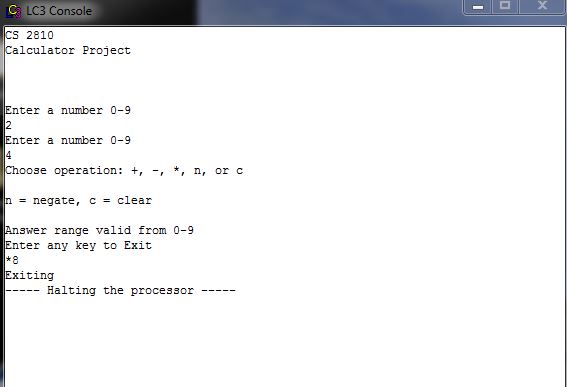
;.FILL #70

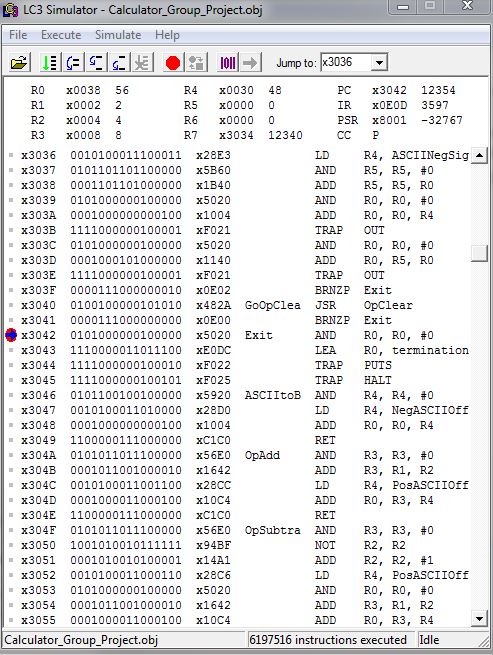
;.FILL #80

;

;Neg10 .FILL xFFF6

.END ;end assembly directive





Screenshots of the LC-3 simulator and console windows of our calculator performing the multiply operation.

Take note of the following registers:

R0 contains the ASCII value of 8

R1 contains the first operand 2

R2 contains the second operand 4

R3 contains the product 8

R4 contains the offset value to convert the binary number into its ASCII equivalent

**5. Conclusions**

From completing this project, we gained a lot of experience with using subroutines. We also gained a good understanding of how a computer does simple arithmetic operations. We attempted to utilize Lookup tables to be able to display double digit numbers between -81 and 81 (10, 20, 30, 40, 50, 60, 70, 80), but we found out that that was significantly complicated, and we were unable to add this feature. Another problem we ran into was running out of room for labels and strings. This caused us to be unable to add features that we desired to include, such as the power and divide subroutines.

**6. References**

Patt, Yale, and Sanjay Patel. *Introduction to Computing Systems, from Bits & Gates to C & beyond*. 2nd ed. New York: McGraw Hill, 2004. 178-288. Print.

Power, D. J., "A Brief History of Spreadsheets", DSSResources.COM, World Wide Web,http://dssresources.com/history/sshistory.html, version 3.6, 08/30/2004. Photo added September 24, 2002.