|  |
| --- |
| **UCF** |
| **EEL 4742 LABORATORY EXPERIMENT #5 Section 12 Andrew Mendez, Daniel Franco 3/18/2014** |
|  |
|  |
| **Student** |
|  |

|  |
| --- |
|  |

**Objective:** To develop a C language and an assembly language program that adds, subtracts and multiplies two hexadecimal digits. The resulting answer is then displayed on the LCD display of the MSP430FG4618 experimenter board.

**Apparatus List:**

* Dell Computer
* Monitor
* Keyboard
* Mouse
* CCS software
* MSP430FG4618

**Procedure and/or Design Methodology:**

This experiment had two separate parts to it, which we will describe in order as we worked on them. All programs should run indefinitely and start on a new line each time waiting for user input. We will provide screenshots for each of the programs we worked on.

Part One, C code:

1. Write a C language program that input from the keyboard two hexadecimal numbers via HyperTerminal. Next these two numbers are to be displayed in the HyperTerminal window and on the right most two digits of the LCD display on the experimenter board.
2. Write a C language program that inputs two ASCII hexadecimal digits followed by the “+” symbol to indicate addition. Next, the user enters a second set of two hexadecimal digits that are to be added to the first two hexadecimal digits. The addition result is shown on both the hyperterminal display as well as the LCD display.
3. Write a C language program that inputs two ASCII hexadecimal digits followed by the “−” symbol to indicate subtraction. Next, the user enters a second set of two hexadecimal digits that are to be subtracted from the first two hexadecimal digits. The subtraction result is shown on both the hyperterminal display as well as the LCD display.
4. Write a C language program that inputs two ASCII hexadecimal digits followed by the “\*” symbol to indicate multiplication. Next, the user enters a second set of two hexadecimal digits that are to be multiplied with the first two hexadecimal digits. The multiplication result is shown on both the hyperterminal display as well as the LCD display.
5. Write a program that merges steps 2, 3, and 4 together and uses the ASCII symbols “+”, “-“, and “\*” to determine which arithmetic operation to perform. This program should run indefinitely and start on a new line each time waiting for user input.

Part Two, Assembly code:

1. Write an assembly language program that inputs from the keyboard two hexadecimal numbers via hyperterminal. Next, these two numbers are displayed in the hyperterminal window and on the right most two digits of the LCD display on the experimenter board.
2. Write an assembly language program that inputs two ASCII hexadecimal digits followed by the “+” symbol to indicate addition. Next, the user enters a second set of two hexadecimal digits that are to be added to the first two hexadecimal digits.
3. Write an assembly language program that inputs two ASCII hexadecimal digits followed by the “-” symbol to indicate subtraction. Next, the user enters a second set of two hexadecimal digits that are to be subtracted from the first two hexadecimal digits. The subtraction result is shown on both the HyperTerminal display as well as the LCD display.
4. Write an assembly language program that inputs two ASCII hexadecimal digits followed by the “\*” symbol to indicate multiplication. Next, the user enters a second set of two hexadecimal digits that are to be multiplied with the first two hexadecimal digits. The multiplication result is shown on both the HyperTerminal display as well as the LCD display.
5. Write a program that merges steps 2, 3, and 4 together and uses the ASCII symbols “+”, “-“, and “\*” to determine which arithmetic operation to perform.

**Design Specification Plan:**

C- Programs:

For the C program part of this experiment, we relied on the code that we worked on during previous laboratory sessions, altered it to complete the specified tasks and added the necessary computations for it to accomplish the designated task. For the first program, we simply combined experiments three and four, while for the second part, after reading in the values, we added them together, printed them on to the board, and converted back to ASCII to be able to print on the HyperTerminal. We followed a very similar procedure for parts three and four.

Assembly programs:

Since the assembly programs were designed to complete the same tasks as the C programs, we followed the same plan, with the difference that this time we would be programming in a different language at a level closer to the hardware. The discussion provided to us in the pre-laboratory assignment helped us thoroughly because it gave us a direct idea of what we should do with the hardware in order to get it to complete the assigned tasks.

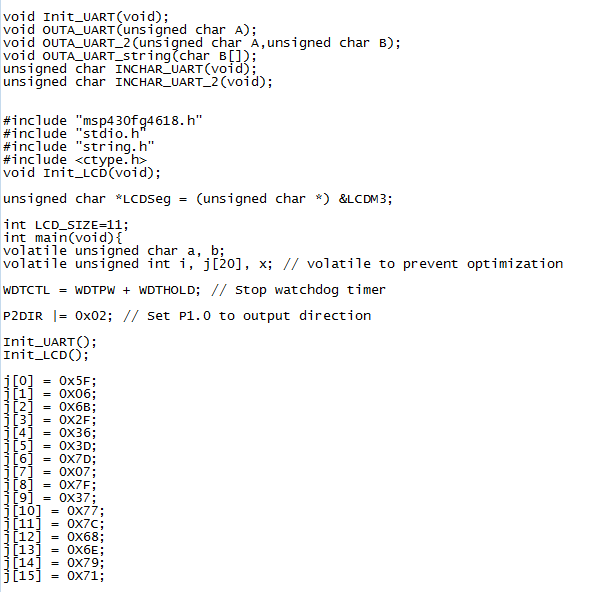
**Test Plan:**

In order to test our code we would be using the MSP-430FG4618 board. By running the code we had created in the Code Composer Studio, we would be able to communicate with the hardware through the Terminal and make sure that the input we had was dealt with correctly and displayed the proper values.

By using the HyperTerminal, we would interact with the board in order to ensure our code was working correctly and it would yield the correct output. This procedure was followed every time we completed a piece of code in any of the languages, C or Assembly.

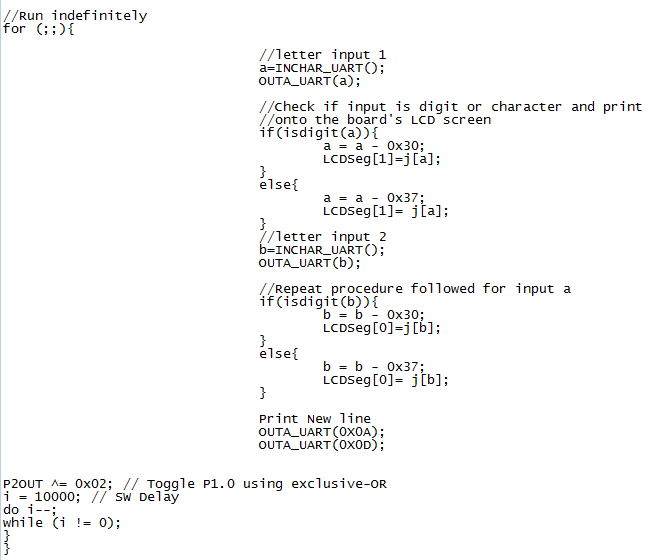
**Source Code:** (We will only be including the primary parts of the codes, if we copy the whole code into here, the report will be too long and we would not want to print too many pages)

For future purposes, this is how we initialized our programs

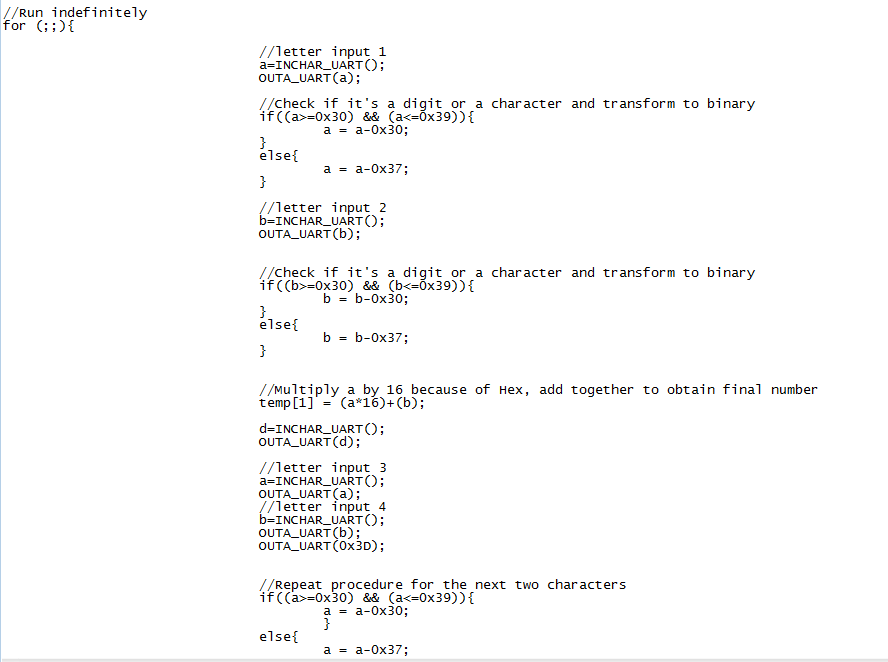


**Part One, C:**

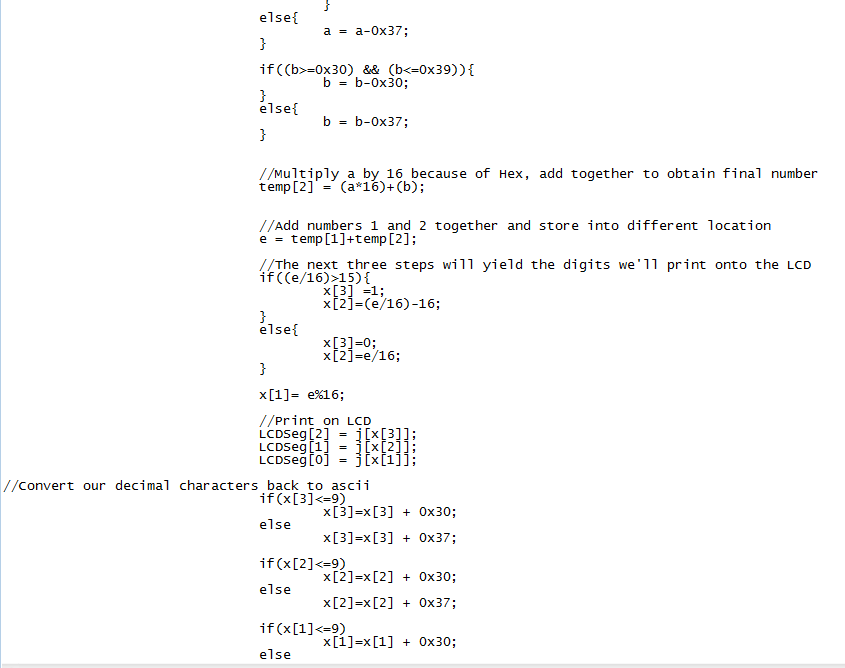
Program 1.



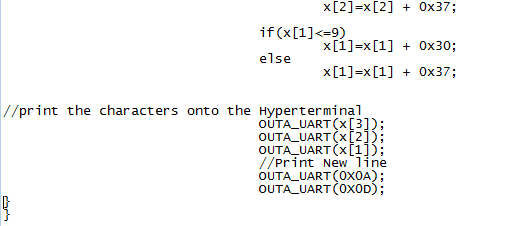
Program 2, Part 1



Program 2, Part 2



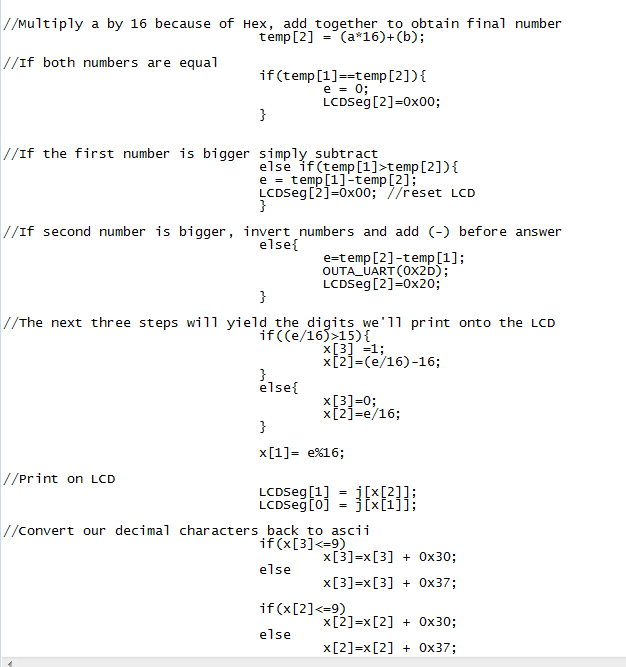
Program 2, Part 3



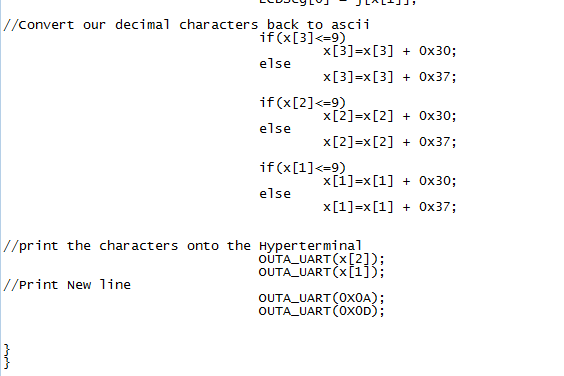
Program 3:

The first part of the code is the same as program 2, it will input two characters or digits and convert them to binary.

Program 3



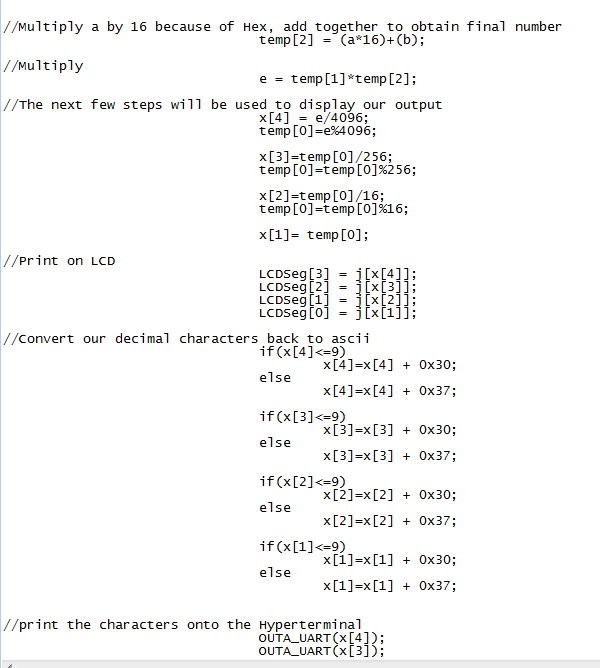
Program 3



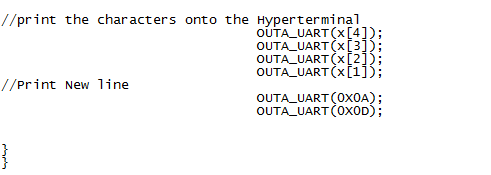
Program 4:

Program 4, as well as program 3 and program 2, starts out by inputting the 4 characters, in sets of 2, and converting them to binary so they can be operated with each other. The following is the code:

Program 4



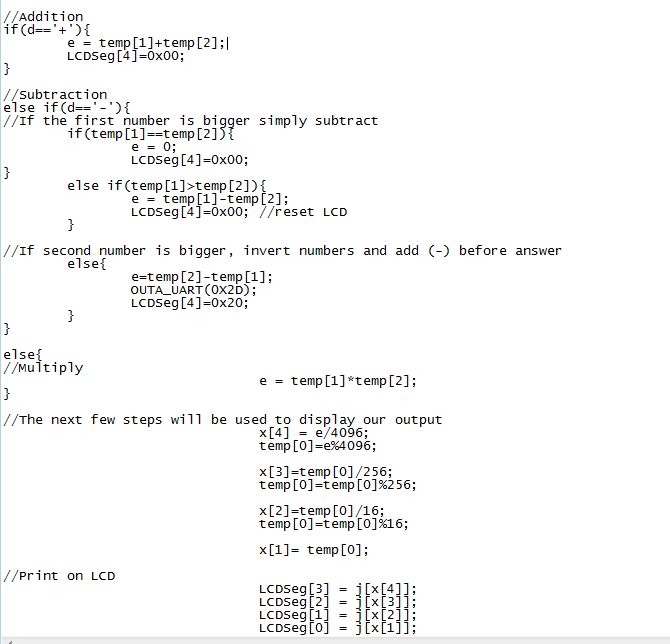
Program 4



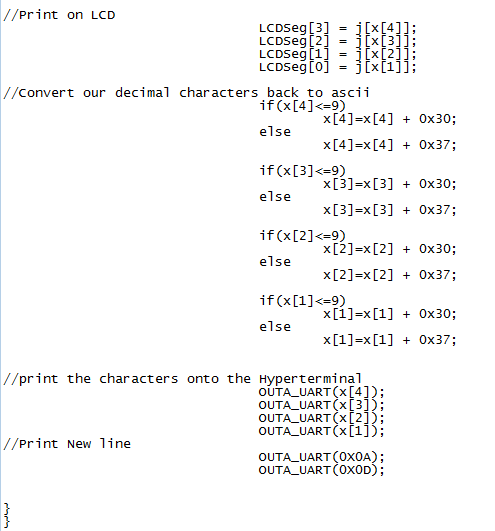
Program 5:

This program, as well as the other 3, starts out by receiving input from the user, so we will not show that part of the code again in order to save space, and paper. There will be a series of ‘if’ statements that will control what happens along the code.

Program 5



Program 5



**Part Two:**

**Program 1:**

Mainloop

call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R11

mov.w #0x0A, R4

call #OUTA\_UART

mov.w #0x0D, R4

call #OUTA\_UART

mov.w #0x2F, R9

cmp.w R10,R9

jge if

cmp.w #0x3A,R10

jge if

sub.w #0x30,R10

if cmp.w R11, R9

jge else

cmp.w #0x3A, R11

jge else

sub.w #0x30,R11

jmp continue

else mov.w #0x40, R9

cmp.w R11,R9

jge continue

cmp.w #0x47, R11

jge continue

sub.w #0x37,R11

continue ;rla.w R10

;rla.w R10

;rla.w R10

;rla.w R10

;add.w R10, R11

mov.w R10,R4

call #OUTA\_UART

mov.w R11,R4

call #OUTA\_UART

mov.w #0x0A, R4

call #OUTA\_UART

mov.w #0x0D, R4

call #OUTA\_UART

call# DISPLAY\_TO\_LCD

jmp Mainloop

DISPLAY\_TO\_LCD

;push.w R5

push.w R6

push.w R7

push.w R8

; R8 is a loop counter to cover all of the segments. This count

; counts up from 0

mov.b #0x00, R8 ;loop counter to go through all numbers

mov.w #LCDM3, R5 ;set up LCD memory pointer to R5

;add.w #0x30,R5

mov.b #0x00, R7 ;move the current number to temp

lpt1

;enter the number 2

; move 0xff into R7 to turn on all LCD segments the LCD memory

;mov.b num(R7), 0(R5) ;send element in num arraty to LCD

and.w #0x00FF,R10

and.w #0x00FF,R11

mov.b num(R11), R14 ;send b to LCD memory for display

mov.b R14,0(R5) ; display first character in

mov.b num(R10), R14 ;send b to LCD memory for display

mov.b R14,1(R5) ; display first character in

;and.b #0x0000,R14

; Increment R5 to point to the next seven segment display

; Increment R8 for the next count inthe loop

;inc.b R8 ;increment R8 for next number in array

inc.b R7

mov.w #0xFFFF,R15 ;Delay to R15

L1

dec.w R15 ; Decrement R15

jnz L1

mov.w #0xFFFF,R15 ;Delay to R15

L2

dec.w R15 ; Decrement R15

jnz L2

;cmp.b 0xLCD\_SIZE, R8 ;check if loop is done

;cmp.w #0x10,R8 ;check if all 16 values are printed

;jnz lpt1 ;if not loop again

;pop.w R5

pop.w R6

pop.w R7

;pop.w R8

ret

**Program 2:**

Mainloop

call #HEX8IN

mov.w R10,R6

mov.w R11,R7

;call# DISPLAY\_TO\_LCD

mov.w #'+',R4 ;add a plus

call #OUTA\_UART

call#HEX8IN

;call# DISPLAY\_TO\_LCD

mov.w #'=',R4 ;add a plus

call #OUTA\_UART

;combine to 8 bit values

rla R6

rla R6

rla R6

rla R6

add.b R6,R7

rla R10

rla R10

rla R10

rla R10

add.b R10,R11

;add together

add.w R7,R11

; swap bytes

;swpb R11

push.w R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R7 ; INDEX FOR lcd TO PRINT

and.w #0X000F,R8

call #HEX10UT ; PRINT 4TH BIT

pop.w R11

push.w R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R8 ; INDEX FOR lcd TO PRINT

and.w #0X000F,R8

call #HEX10UT ;PRINT 3RD BIT

pop.w R11

mov.b R11,R9 ; INDEX FOR lcd TO PRINT

and.w #0X000F,R9

call #HEX10UT ; PRINT LAST BIT

call #DISPLAY\_TO\_LCD\_CHAR

mov.b #0X0A,R4

call #OUTA\_UART

mov.w #0X0D,R4

call #OUTA\_UART

jmp Mainloop

HEX10UT ; this function displays a 4 bit character

push.w R11

and.w #0x000F,R11

cmp.b #0x0A, R11

jhs Letter2

add.b #0x30, R11

mov.b R11,R4

call #OUTA\_UART

jmp LP2

Letter2 add.b #0x37,R11

mov.b R11,R4

call #OUTA\_UART

LP2 pop.w R11

ret

HEX8IN call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #Convert

mov.w R10,R8

call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #Convert

mov.w R10, R11 ; store 1st character

mov.w R8, R10; store second character

ret

;mov.w R11,R4

;call #OUTA\_UART

DISPLAY\_TO\_LCD\_CHAR ; INPUT R4-LAST CHAR PRINTED IN HYPERTERMINAL

mov.w #LCDM3, R5 ;set up LCD memory pointer to R5

;add.w #0x30,R5

;mov.b #0x00, R7 ;move the current number to temp

lpt1

;enter the number 2

; move 0xff into R7 to turn on all LCD segments the LCD memory

;mov.b num(R7), 0(R5) ;send element in num arraty to LCD

;and.w #0x000F,R10

;and.w #0x000F,R11

mov.b num(R7), R7 ;send b to LCD memory for display

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display first character in

mov.b R7 , 2(R5) ; display first character in

;and.b #0x0000,R14

; Increment R5 to point to the next seven segment display

; Increment R8 for the next count inthe loop

;inc.b R8 ;increment R8 for next number in array

;inc.b R7

mov.w #0xFFFF,R15 ;Delay to R15

L1

dec.w R15 ; Decrement R15

jnz L1

mov.w #0xFFFF,R15 ;Delay to R15

L2

dec.w R15 ; Decrement R15

jnz L2

;cmp.b 0xLCD\_SIZE, R8 ;check if loop is done

;cmp.w #0x10,R8 ;check if all 16 values are printed

;jnz lpt1 ;if not loop again

;pop.w R5

;pop.w R6

;pop.w R7

;pop.w R8

ret

**Program 3:**

; Main Code

LCD\_SIZE .byte 11 ; eleven bytes needed by the LCD

;Array of hex values that coorespond the correct 0-F display on LCD

num .byte 0x5F,0x06,0x6B,0x2F,0x36

.byte 0x3D,0x7D,0x07,0x7F,0x3F

.byte 0x77,0x07C,0x59,0x6E,0x79,0x71

;----------------------------------------------------------------------

;----------------------------------------------------------------

.text ; program start

.global \_START ; define entry point

;----------------------------------------------------------------

START mov.w #300h,SP ; Initialize 'x1121

; stackpointer

StopWDT mov.w #WDTPW+WDTHOLD,&WDTCTL ; Stop WDT

call #Init\_UART

call #Init\_LCD ; go initialize the LCD Display

main:

;; call #INCHAR\_UART

;; mov.w #0x2D,R4 add a plus

;; call #OUTA\_UART

; call #uartGetChar

;; call #switchCase

;; call #uartPutChar

;; xor.b #0x41 , &P1OUT

call #HEX8IN ; get first number

mov.w R10,R6

mov.w R11,R7

;call# DISPLAY\_TO\_LCD

mov.w #0x2D,R4 ;add a minus

call #OUTA\_UART

call #HEX8IN ;get second character

;call# DISPLAY\_TO\_LCD

mov.w #0x3D,R4 ;add a equal

call #OUTA\_UART

;combine to 8 bit values

rla R6

rla R6

rla R6

rla R6

add.b R6,R7

rla R10

rla R10

rla R10

rla R10

add.b R10,R11

;subtract together

;compare two values to subtract bigger num from smaller num

cmp.b R7,R11

jhs SUB1 ;jump if R7 > R11

; else R7< R11

sub.w R11,R7

mov.w R7,R11 ; store result in r11 register to be printed

mov.b #0x00,R7 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R7

jmp PRINT

; swap bytes

SUB1: sub.w R7,R11

mov.b #0xFF,R7 ; INDEX FOR lcd TO PRINT

mov.w #0x2D,R4 ;add a minus

call #OUTA\_UART; PRINT NEGATIVE SIGN

;swpb R11

PRINT: ;print the resulting 8 bits

push.w R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R8 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R8

call #HEX10UT ;PRINT 3RD BIT

pop.w R11

mov.b R11,R9 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R9

call #HEX10UT ; PRINT LAST BIT

call #DISPLAY\_TO\_LCD\_CHAR

mov.b #0x0A,R4

call #OUTA\_UART

mov.w #0x0D,R4

call #OUTA\_UART

jmp main

HEX10UT ; this function displays a 4 bit character

push.w R11

and.w #0x000F,R11

cmp.b #0x0A, R11

jhs Letter2

add.b #0x30, R11

mov.b R11,R4

call #OUTA\_UART

jmp LP2

Letter2 add.b #0x37,R11

mov.b R11,R4

call #OUTA\_UART

LP2 pop.w R11

ret

HEX8IN call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #Convert

mov.w R10,R8

call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #Convert

mov.w R10, R11 ; store 1st character

mov.w R8, R10; store second character

ret

;mov.w R11,R4

;call #OUTA\_UART

DISPLAY\_TO\_LCD\_CHAR ; INPUT R4-LAST CHAR PRINTED IN HYPERTERMINAL

mov.w #LCDM3, R5 ;set up LCD memory pointer to R5

;add.w #0x30,R5

;mov.b #0x00, R7 ;move the current number to temp

lpt1

;enter the number 2

; move 0xff into R7 to turn on all LCD segments the LCD memory

;mov.b num(R7), 0(R5) ;send element in num arraty to LCD

;and.w #0x000F,R10

;and.w #0x000F,R11

cmp.b #0xFF, R7

jeq PRINT\_NEG

mov.b num(R7), R7 ;send b to LCD memory for display

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display first character in

mov.b R7 , 2(R5) ; display first character in

jmp DELAY

PRINT\_NEG

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display first character in

mov.b #0x20 , 2(R5) ; display first character in

;and.b #0x0000,R14

; Increment R5 to point to the next seven segment display

; Increment R8 for the next count inthe loop

;inc.b R8 ;increment R8 for next number in array

;inc.b R7

DELAY mov.w #0xFFFF,R15 ;Delay to R15

L1

dec.w R15 ; Decrement R15

jnz L1

mov.w #0xFFFF,R15 ;Delay to R15

L2

dec.w R15 ; Decrement R15

jnz L2

;cmp.b 0xLCD\_SIZE, R8 ;check if loop is done

;cmp.w #0x10,R8 ;check if all 16 values are printed

;jnz lpt1 ;if not loop again

;pop.w R5

;pop.w R6

;pop.w R7

;pop.w R8

ret

**Program 4:**

main:

;; call #INCHAR\_UART

;; mov.w #0x2D,R4 add a plus

;; call #OUTA\_UART

; call #uartGetChar

;; call #switchCase

;; call #uartPutChar

;; xor.b #0x41 , &P1OUT

call #HEX8IN ; get first number

mov.w R10,R6

mov.w R11,R7

;call# DISPLAY\_TO\_LCD

mov.w #'\*',R4 ;add a minus

call #OUTA\_UART

call #HEX8IN ;get second character

;call# DISPLAY\_TO\_LCD

mov.w #'=',R4 ;add a equal

call #OUTA\_UART

;combine to 8 bit values

rla R6

rla R6

rla R6

rla R6

add.b R6,R7

rla R10

rla R10

rla R10

rla R10

add.b R10,R11

mov.w #0x0000,R9

;multiply together R7 \* R11

Multiply:

;flag to determine what to print

mov.b #0xFE,R13 ; INDEX FOR lcd TO PRINT

cmp.w #0x0000,R11

jeq PRINT; print if equal to zero, end loop

add.w R7, R9

dec R11

jmp Multiply

PRINT: ;print the resulting 8 bits

mov.w R9,R11

push.w R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R6 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R6

call #HEX10UT ;PRINT 4TH HEX

pop.w R11

push.w R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R7 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R7

call #HEX10UT ;PRINT 3RD HEX

pop.w R11

push.w R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R8 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R8

call #HEX10UT ;PRINT 2ND HEX

pop.w R11

mov.b R11,R9 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R9

call #HEX10UT ; PRINT LAST HEX

call #DISPLAY\_TO\_LCD\_CHAR

mov.b #0x0A,R4

call #OUTA\_UART

mov.w #0x0D,R4

call #OUTA\_UART

jmp main

HEX10UT ; this function displays a 4 bit character

push.w R11

and.w #0x000F,R11

cmp.b #0x0A, R11

jhs Letter2

add.b #0x30, R11

mov.b R11,R4

call #OUTA\_UART

jmp LP2

Letter2 add.b #0x37,R11

mov.b R11,R4

call #OUTA\_UART

LP2 pop.w R11

ret

HEX8IN call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #Convert

mov.w R10,R8

call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #Convert

mov.w R10, R11 ; store 1st character

mov.w R8, R10; store second character

ret

;mov.w R11,R4

;call #OUTA\_UART

DISPLAY\_TO\_LCD\_CHAR ; INPUT R4-LAST CHAR PRINTED IN HYPERTERMINAL

mov.w #LCDM3, R5 ;set up LCD memory pointer to R5

;add.w #0x30,R5

;mov.b #0x00, R7 ;move the current number to temp

lpt1

;enter the number 2

; move 0xff into R7 to turn on all LCD segments the LCD memory

;mov.b num(R7), 0(R5) ;send element in num arraty to LCD

;and.w #0x000F,R10

;and.w #0x000F,R11

cmp.b #0xFF, R7

jeq PRINT\_NEG

mov.b num(R6), R6 ;send b to LCD memory for display

mov.b num(R7), R7 ;send b to LCD memory for display

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display first character in

mov.b R7 , 2(R5) ; display first character in

mov.b R6 , 3(R5) ; display first character in

jmp DELAY

PRINT\_NEG

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display first character in

mov.b #0x20 , 2

;and.b #0x0000,R14

; Increment R5 to point to the next seven segment display

; Increment R8 for the next count inthe loop

;inc.b R8 ;increment R8 for next number in array

;inc.b R7

DELAY mov.w #0xFFFF,R15 ;Delay to R15

L1

dec.w R15 ; Decrement R15

jnz L1

mov.w #0xFFFF,R15 ;Delay to R15

L2

dec.w R15 ; Decrement R15

jnz L2

;cmp.b 0xLCD\_SIZE, R8 ;check if loop is done

;cmp.w #0x10,R8 ;check if all 16 values are printed

;jnz lpt1 ;if not loop again

;pop.w R5

;pop.w R6

;pop.w R7

;pop.w R8

ret

**Progam 6:**

main:

;; call #INCHAR\_UART

;; mov.w #0x2D,R4 add a plus

;; call #OUTA\_UART

; call #uartGetChar

;; call #switchCase

;; call #uartPutChar

;; xor.b #0x41 , &P1OUT

call #HEX8IN ; get first number

mov.w R10,R6

mov.w R11,R7

;call# DISPLAY\_TO\_LCD

call #INCHAR\_UART ; get operation

mov.w R4, R12 ; store the operation to check

call #OUTA\_UART

call #HEX8IN ;get second character

;call# DISPLAY\_TO\_LCD

mov.w #'=',R4 ;add a equal

call #OUTA\_UART

;combine to 8 bit values

rla R6

rla R6

rla R6

rla R6

add.b R6,R7

rla R10

rla R10

rla R10

rla R10

add.b R10,R11

;multiply together R7 \* R11

mov.w R11, R8

mov.w #0x0000,R9

;determine operation

cmp.b #'\*', R12

jeq Multiply

cmp.b #'+', R12

jeq Addition ;jeq Addition

cmp.b #'-', R12

jeq Subtraction ;jeq Addition

Addition:

;add together

mov.b #0xF0,R13 ; INDEX FOR lcd TO PRINT

add.w R7,R11

mov.w R11,R9 ;move result to prepare printing

jmp PRINT

Subtraction: ;subtract together

mov.b #0x00,R13 ; INDEX FOR lcd TO PRINT

;compare two values to subtract bigger num from smaller num

cmp.b R7,R11

jhs SUB1 ;jump if R7 > R11

; else R7< R11

sub.w R11,R7

;mov.w R7,R11 ; store result in r11 register to be printed

;mov.b #0x00,R7 ; INDEX FOR lcd TO PRINT

;and.w #0x000F,R7

mov.w R7,R9 ; move answer to print

jmp PRINT

; swap bytes

SUB1: sub.w R7,R11

;flag to determine what to print

mov.b #0xFF,R13 ; INDEX FOR lcd TO PRINT

mov.w #0x2D,R4 ;add a minus

call #OUTA\_UART; PRINT NEGATIVE SIGN

mov.w R11,R9 ; move answer to print

jmp PRINT

Multiply:

;flag to determine what to print

mov.b #0xFE,R13 ; INDEX FOR lcd TO PRINT

cmp.w #0x0000,R11

jeq PRINT; print if equal to zero, end loop

add.w R7, R9

dec R11

jmp Multiply

PRINT: ;print the resulting 8 bits

mov.w R9,R11

push.w R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R6 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R6

call #HEX10UT ;PRINT 4TH HEX

pop.w R11

push.w R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R7 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R7

call #HEX10UT ;PRINT 3RD HEX

pop.w R11

push.w R11

rra R11

rra R11

rra R11

rra R11

mov.b R11,R8 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R8

call #HEX10UT ;PRINT 2ND HEX

pop.w R11

mov.b R11,R9 ; INDEX FOR lcd TO PRINT

and.w #0x000F,R9

call #HEX10UT ; PRINT LAST HEX

call #DISPLAY\_TO\_LCD\_CHAR

mov.b #0x0A,R4

call #OUTA\_UART

mov.w #0x0D,R4

call #OUTA\_UART

mov.w #0x0000,R6

mov.w #0x0000,R7

mov.w #0x0000,R8

mov.w #0x0000,R9

jmp main

HEX10UT ; this function displays a 4 bit character

push.w R11

and.w #0x000F,R11

cmp.b #0x0A, R11

jhs Letter2

add.b #0x30, R11

mov.b R11,R4

call #OUTA\_UART

jmp LP2

Letter2 add.b #0x37,R11

mov.b R11,R4

call #OUTA\_UART

LP2 pop.w R11

ret

HEX8IN call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #Convert

mov.w R10,R8

call #INCHAR\_UART

call #OUTA\_UART

mov.w R4, R10

call #Convert

mov.w R10, R11 ; store 1st character

mov.w R8, R10; store second character

ret

;mov.w R11,R4

;call #OUTA\_UART

DISPLAY\_TO\_LCD\_CHAR ; INPUT R4-LAST CHAR PRINTED IN HYPERTERMINAL

mov.w #LCDM3, R5 ;set up LCD memory pointer to R5

;add.w #0x30,R5

;mov.b #0x00, R7 ;move the current number to temp

lpt1

;enter the number 2

; move 0xff into R7 to turn on all LCD segments the LCD memory

;mov.b num(R7), 0(R5) ;send element in num arraty to LCD

;and.w #0x000F,R10

;and.w #0x000F,R11

cmp.b #0xFF, R13

jeq PRINT\_NEG

cmp.b #0xFE, R13

jeq PRINT\_MULT

cmp.b #0xF0,R13

jeq PRINT\_ADD

cmp.b #0x00,R13

jeq PRINT\_SUB2

;mov.b num(R7), R7 ;send b to LCD memory for display

PRINT\_ADD:

;mov.b num(R7), R6 ;send b to LCD memory for display

mov.b num(R7), R7 ;send b to LCD memory for display

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display second character in

mov.b R7 , 2(R5) ; display third character in

;mov.b R6 , 3(R5) ; display third character in

jmp DELAY

PRINT\_NEG:

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display first character in

mov.b #0x20 , 2(R5) ; display first character in

jmp DELAY

PRINT\_SUB2:

mov.w #0x0000,R7

mov.b num(R7), R7 ;send b to LCD memory for display

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display first character in

mov.b R7 , 2(R5) ; display first character in

jmp DELAY

PRINT\_MULT:

mov.b num(R6), R6 ;send b to LCD memory for display

mov.b num(R7), R7 ;send b to LCD memory for display

mov.b num(R8), R8 ;send b to LCD memory for display

mov.b num(R9), R9 ;send b to LCD memory for display

mov.b R9 , 0(R5) ; display first character in

mov.b R8 , 1(R5) ; display second character in

mov.b R7 , 2(R5) ; display third character in

mov.b R6 , 3(R5) ; display third character in

jmp DELAY

;and.b #0x0000,R14

; Increment R5 to point to the next seven segment display

; Increment R8 for the next count inthe loop

;inc.b R8 ;increment R8 for next number in array

;inc.b R7

DELAY mov.w #0xFFFF,R15 ;Delay to R15

L1

dec.w R15 ; Decrement R15

jnz L1

mov.w #0xFFFF,R15 ;Delay to R15

L2

dec.w R15 ; Decrement R15

jnz L2

;cmp.b 0xLCD\_SIZE, R8 ;check if loop is done

;cmp.w #0x10,R8 ;check if all 16 values are printed

;jnz lpt1 ;if not loop again

;pop.w R5

;pop.w R6

;pop.w R7

;pop.w R8

ret

**Conclusion:** In this lab, we devised a Hexadecimal Calculator using the MSP430 Experimenter Board. We developed a routine to add, subtract, and multiply two hexadecimal numbers and display the result on the LCD. To develop these routines, we programmed the routines using a C language and an native assembly language on the MSP430. The resulting answer is then displayed on the LCD display of the MSP430FG4618 experimenter board.