Microprocessors Final Lab Project

**Author: *Michael Jordan Sallmen***

**Introductory lab session for the lab project (Written Summary begins on pg. 5-6, and source code on pg. 7-14)**

**Objectives:**

* Implementing the knowledge of Assembly and microcontroller.
* Learning the use of different hardware capabilities of the Dragon EVB.

**Introduction:**

This lab session is an introductory lab session for the lab project, students are supposed to follow the guidelines and learn the use of different hardware attached to the Dragon EVB. Students will be making use of this hardware in their project. The project tasks and objectives will be provided during the next lab session.

**Project Rules**

* The lab project counts for 35% of lab grade.
* This project is to be done in a group of two students, during the start of this lab session each group is supposed to submit the names of their group members.
* Performance of groups in the project will be only their own, students are not allowed to copy the code and strategies used by other groups.
* Students can ask the TA for help but the help provided will be very limited, as this project is the final evaluation for this lab.
* Students can download the Code Warrior on their personal computers and work at home or off campus. They can test their codes on the Dragon EVB during the regular lab sessions.
* If any students want to use the lab facility other than lab sessions, he/she can contact the TA during office hours.
* Projects will be graded during the last lab session.
* 5 % credit is for a project report which will also be collected on the final lab day, TA will explain the format for the lab report during the lab session.
* Remember Google is your best friend and Book is the best resource.

**Seven Segment Display**

A seven segment display is used for displaying digits. Decimal and Hex digits can be displayed on a seven segment by turning on the appropriate segments. Following example from your book demonstrates the use of Seven segment display. Follow it and try to understand the working of a seven segment display.

**Interfacing with Seven-Segment Displays**

* Seven-segment displays are mainly used to display decimal digits and a small set of letters.
* The HCS12 I/O port can drive a seven-segment display directly.
* Buffer chips are used mainly to save excessive current draw from the HCS12.
* An example circuit for interfacing with seven-segment display is shown in Figure 4.17
* The segment A, B, C, D, E, F, G and Decimal Point (Dragon EVB LCD also has a decimal point) are driven by PB0, PB1, PB2, PB3, PB4, PB5, PB6 and PB7, respectively.
* The microcontroller must send an appropriate value to the output in order to display certain value.
* The naming convention of the segments from a to g is universal.

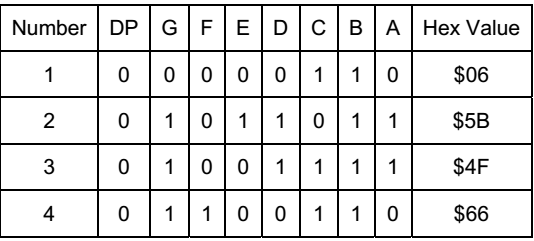


Following table shows the sequence to be sent by the microcontroller in order to turn the respective displays on.

|  |  |
| --- | --- |
| **Display** | **Value on Port P** |
| 3 (left most) | 0E |
| 2 | 0D |
| 1 | 0B |
| 0 (right most) | 07 |

The seven segment displays are common cathode, so in order to turn each display on you need to send a zero to the concerned pin of port P. For example to turn the display 3 on you need to send 0E to port P. The code in the second column is to be sent to PORT P in order to turn the display in first column on.

The following table contains the code for displaying the first four decimal digits on the seven segment displays, you can send these values to PORT B to display the corresponding digits.



If you send the value &06 to the PORT B you will see number 1 appearing on the display, students are encouraged to complete the table themselves.

The following program displays the digit 6 on the display 3 of Dragon EVB. Study it closely and try displaying other digits on different displays. Also study the example 4.14 form your book and display multiple digits on the seven segment using the technique of **time multiplexing**.

LDAA #$FF

STAA DDRB

LDAA #$FF

STAA DDRP

LDAA #$1F

STAA PTP

LDAA #$7D

STAA PTB

You can play around with this code and combine it with the delay subroutine from your last lab. Make sure that you know how to use the seven segment display as it will form an integral part of your project.

**DIP Switches (Input)**

The Port H of HCS12 microcontroller is connected to an 8-position DIP switch. The DIP switch is connected to GND via the eight 4.7K resistors, so it’s not dead short to GND. Following diagram has been taken from your book but it is not very accurate as according to the manual of HCS12 the resisters are of value 4.7K. On our level the value of resistors does not matter.



Following lines of code can be used to collect input from the DIP switches. You can verify the data collected by displaying it on the PORT B LEDs or on the seven segment display.

LDAA #$0

STAA DDRH

LDAA PTH

Loading 0 in PORT H makes it an input port. And the last line of code collects the input data from PORT H and stores it into register A.

**Note:**

* **The hardware covered in this manual are just to introduce you to the hardware capabilities of the Dragon EVB. It is not necessary that you use only this hardware for your project. You can make use of the LCD, on board KEYBOARD or any other I/O port to accomplish the task or for adding beauty to it.**
* **You are not supposed to submit any lab report for this lab, this lab is a part of the lab project. Work in groups and do not discuss with other groups.**

**Solution to Lab Final Project**

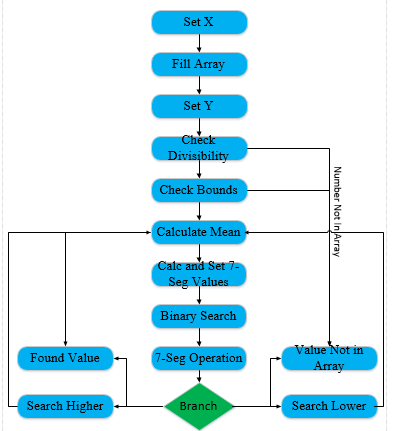
**Abstract:**

This project is the result of the final lab project that was assigned. The program written is to take user input as x value, fill array with length of 50 numbers. A y value is entered and a binary search function runs searching for that value in the array looking for midpoint value in each iteration. During each iteration, the seven-segment display shows the current mean value for approximately 2 seconds. For the second part of the project, DIP switches are to be used to initialize x and y at runtime, rather than statically.

**Introduction:**

This project was an interesting and relatively difficult one.  It tested our knowledge on assembly in a broad sense and brought together what we had learned throughout the semester effectively.  We were able to utilize our skills with loops, allocating memory, time delay, and register transfer and arithmetic.  These will be shown in detail in the conclusion and via comments located in our code.

**Conclusion:**  
We were able to completely finish Part A but could not figure out Part B.  In A we started by initializing the array utilizing our index registers which we had little trouble figuring out.  Then we continued with the binary search, the search itself led to minor difficulties and we were able to get it to work for all values with a couple of separate “check” loops.  Once the binary search was completed the task of displaying them on the 7 segment display was next; here we had some difficulty due to our lack of experience with the display.  Our first goal was to get the time delay between each digit correct which was relatively simple, but the 2 second delay for each number in the search proved more difficult.  At first we put the display function itself in a loop to last 2 seconds; this led to issues on the display and we also realized that different numbers would make the display loop take more clock cycles and thus more time.  Since our first idea became problematic we changed our code so that the display function would just call the delay when needed rather than having it incorporated into the delay.  This solved our issues and Part A was complete.

**Flow Chart:**

**Assembly code starts below this line:**

ORG $4000 ;Flash ROM address for Dragon12+

LDS #$4000

Entry:

delayCount EQU $3500

segOne EQU $3000 ;Leftmost segment

LDAA #$3F

STAA segOne ;Stores 0 output into left display

segTwo EQU $3002 ;Second

segThree EQU $3004 ;Third

segFour EQU $3006 ;Rightmost segment

disMem EQU $3010 ;7-segment output table

LDX #$3010

LDAA #$3F

STAA 0,X

LDAA #$06

STAA 1,X

LDAA #$5B

STAA 2,X

LDAA #$4F

STAA 3,X

LDAA #$66

STAA 4,X

LDAA #$6D

STAA 5,X

LDAA #$7D

STAA 6,X

LDAA #$33

STAA 7,X

LDAA #$7F

STAA 8,X

LDAA #$6F

STAA 9,X

LDAA #$77 ;A

STAA 10,X

LDAA #$7C ;B

STAA 11,X

LDAA #$39 ;C

STAA 12,X

LDAA #$5E ;D

STAA 13,X

LDAA #$79 ;E

STAA 14,X

LDAA #$71 ;F

STAA 15,X

COUNT EQU $2500

LDAB #50

STAB COUNT

CLRB

Xval EQU $2000

LDD #10 ;Set Xval here: (Value that is used in array creation)

STD Xval ;D is stored in Xval

LDX #$2000 ;Used to hold array offset + starting point

LDY #$2000 ;Used to hold array minimum address

fillArray ;Fills the array

DEC COUNT ;Decreases

STD 0,X

INX ;Increments index register

INX

ADDD Xval ;Adds (value in D) + Xval

TST COUNT ;Compares COUNT to zero

LBGT fillArray ;Branches if COUNT is greater that zero

CLRA

CLRB

Yval EQU $1606

LDD #69 ;Set Yval here: (Value that we are searching the array for)

STD Yval

Max EQU $1600 ;Max bound

STX Max

Min EQU $1602 ;Min bound

STY Min

Mean EQU $1604 ;Average (Midpoint)

LDD Yval ;Checks for values not divisible by Xval

TFR X,Y

LDX Xval

IDIV

CPD #0

LBNE valueNotInArray

LDD Max ;Checks to see if Yval is below the Min or above the Max value of the array

TFR D,X

DEX ;The DEX instructions moves the location from the maximum bound, to two bytes before it

DEX

LDD X

CPD Yval

LBLO valueNotInArray

LDD Min

TFR D,X

LDD X

CPD Yval

LBGT valueNotInArray

TFR Y,X

meanCalc

CPD Yval

LBEQ foundValue

CLRA

CLRB

LDD Min ;Loads minimum value into D

ADDD Max

LSRD ;Divides D by 2

fixByte

TFR D,X ;Stores the memory location in D to X

LDD X ;Loads D with the value in memory location X

CMPB #0

LBEQ check

CMPB #1

LBEQ check

STD Mean ;Stores D value in mean

byteReturn

LDY #$3010

CMPA #$1

LBNE skip

INY

LDD Y

STAA segTwo ;Sets 7-seg digit 2

JMP ab

skip

LDY #$3010

LDD Y

STAA segTwo

ab ;Sets 7-seg digits 3 and 4

init EQU $2506

LDD Mean

CMPB #15 ;Sets 7-seg digits 3 and 4 if less than 15

LBHI set

LDAA #$3F

STAA segThree

LDY #$3010

ABY

LDD Y

STAA segFour

JMP binarySearch

set ;Case if greater than 15

LDD Mean

temp EQU $1100 ;Non-shifted B

STAB temp

LSRB

LSRB

LSRB

LSRB

tempB EQU $1150

STAB tempB ;Shifted B

LDY #$3010

ABY

LDD Y

STAA segThree

LDAA #16

LDAB tempB

MUL ;Multiply B and A

tot EQU $1200

STD tot

LDD Mean

SUBD tot ;Subtracts to retrieve rightmost digit

LDY #$3010

ABY

LDD Y

STAA segFour

JMP binarySearch

binarySearch ;Binary Search function that traverses the array to find the y-value input

JMP sevenSegmentDisplay

binRet

LDD Mean

CPD Yval ;Compares D(contains mean) to the y-value(number searching for)

LBLO searchHigher ;The mean is less than the y-value being searched for

LBGT searchLower ;The mean is greater than the y-value being searched for

LBEQ foundValue ;Current mean value is the y-value, moves out of binary search

JMP valueNotInArray ;Value is not found in Array, jumps out of binary search

searchHigher

INX

INX

STX Min ;Sets Minimum bound

JMP meanCalc

searchLower

DEX

DEX

STX Max ;Sets Maximum bound

JMP meanCalc

check

DEX

LDD X

STD Mean

JMP byteReturn

foundValue

LDAA #$FF ;set-up for 7-segment

STAA DDRB

LDAA #$FF

STAA DDRP

LDAA #$0E ;Left Most Digit

STAA PTP

LDAA segOne

STAA PORTB

JSR delayMS

LDAA #$0D ;Second From Left Digit

STAA PTP

LDAA segTwo

STAA PORTB

JSR delayMS

LDAA #$0B ;Third From Left Digit

STAA PTP

LDAA segThree

STAA PORTB

JSR delayMS

LDAA #$07 ;Right Most Digit

STAA PTP

LDAA segFour

STAA PORTB

JSR delayMS

JMP foundValue ;Infinite loop for found value

valueNotInArray

loopInfinite

LDAA #$FF ;set-up for 7-segment

STAA DDRB

LDAA #$FF

STAA DDRP

LDAA #$0E ;Left Most Digit

STAA PTP

LDAA #$3F

STAA PORTB

JSR delayMS

LDAA #$0D ;Second From Left Digit

STAA PTP

LDAA #$79

STAA PORTB

JSR delayMS

LDAA #$0B ;Third From Left Digit

STAA PTP

LDAA #$33

STAA PORTB

JSR delayMS

LDAA #$07 ;Right Most Digit

STAA PTP

LDAA #$33

STAA PORTB

JSR delayMS

JMP loopInfinite

sevenSegmentDisplay ;7-segment display loop

LDAA #100 ;Loop time control: Higher value will operate loop longer

STAA delayCount

CLRA

CLRB

loop1

LDAA #$FF ;set-up for 7-segment

STAA DDRB

LDAA #$FF

STAA DDRP

LDAA #$0E ;Displays Left Most Digit

STAA PTP

LDAA segOne

STAA PORTB

JSR delayMS

LDAA #$0D ;Display Second From Left Digit

STAA PTP

LDAA segTwo

STAA PORTB

JSR delayMS

LDAA #$0B ;Display Third From Left Digit

STAA PTP

LDAA segThree

STAA PORTB

JSR delayMS

LDAA #$07 ;Displays Right Most Digit

STAA PTP

LDAA segFour

STAA PORTB

JSR delayMS

DEC delayCount

LDAA delayCount

CMPA #0

LBNE loop1

LBRA binRet

delayMS ;1ms delay function

CLRA

LDD #16000

MEMO EQU $1000

STD MEMO

CLRA

CLRB

JSR DELAY

RTS

ORG $6000

DELAY

PSHD

PULD

INCA

DECA

ADDD #2

CPD MEMO

BLT DELAY

RTS

end

END