ECE 212 Lab - Introduction to Microprocessors
Department of Electrical and Computer Engineering
University of Alberta

Lab 1: Introduction to Assembly Language.

Student Name	Student ID
Marlene Gong	1572719
Darius Fang	1570975

Table of Contents Page

1. **Introduction…**……………………………………………………………………………ii
2. **Design**
   Part A…………………………………………………………………………………1
   Part B…………………………………………………………………………………1
3. **Testing**
   Part A…………………………………………………………………………………2
   Part B…………………………………………………………………………………3
4. **Questions**…………………………………………………………………………4
5. **Conclusion**………………………………………………………………………4
6. **Appendix**…………………………………………………………………………5
7. **Marking Sheet**……………………………………………………………………10

Introduction

In Part A of this lab, the goal was to convert an ASCII character to its decimal equivalent. Valid characters were 0-9, A-F, and a-f. When converted, each value would be stored in a memory location. Any other ASCII characters would have the Error Code 0xFFFFFFFF stored in the memory location instead. The start address in which data was stored at 0x43000000 and the start address where converted values were stored at 0x43100000. The program would exit if it came across a memory location in which the content contained the Enter Code 0x0D, thus signalling that there would be no more conversions to be done.

Part B required us to convert an ASCII letter to its upper/lower case equivalent. Valid characters included all of the letters of the alphabet. Similar to Part A, any other characters would have the Error Code stored instead. The start address where data was stored was the same as Part A while the start address for converted values became 0x43200000. The Enter Code would once again signal the end of the program.

The overall objective of this lab was to have students get familiar with developing code in assembly language using basic instructions.

Design

Part A

The program starts by obtaining the value to be converted. It outputs the error code 0xFFFFFFFF if the value is less than 0x30, between 0x3A to 0x3F, or between 0x47 to 0x60, or greater than 0x67. If the value is the same as the exit value, then the program will no longer loop, otherwise the value will be converted to the decimal value. For example 'A' ascii value will be converted to 10, '2' ASCII value will be converted to the decimal value 2. The starting memory location to obtain the values that are going to be converted is at memory address number 0x43000000, and the output memory address to send will be at memory address 0x43100000, both with intervals of 4 hex values (1 longword size).

Sample calculation of conversion from ASCII to decimal values

eg. ASCII ' A ' or 0x41
=> 0d65
- Uppercase, therefore subtract by 0d55
  => 65 - 55 = 10
  Decimal value of A is 10.

Part B

Similarly to part A, it outputs the error code 0xFFFFFFFF if the value is less than 0x41, between 0x5B to 0x60, or greater than 0x7B. It will convert the values to lower case if the value is between 0x41 to 0x5A. Otherwise it will convert the values to uppercase from 0x61 to 0x7A. The code loops until the exit code is obtained. The starting memory location to obtain the values that are going to be converted is at memory address number 0x43000000, and the output memory address to send will be at memory address 0x43200000, both with intervals of 4 hex values (1 longword size).

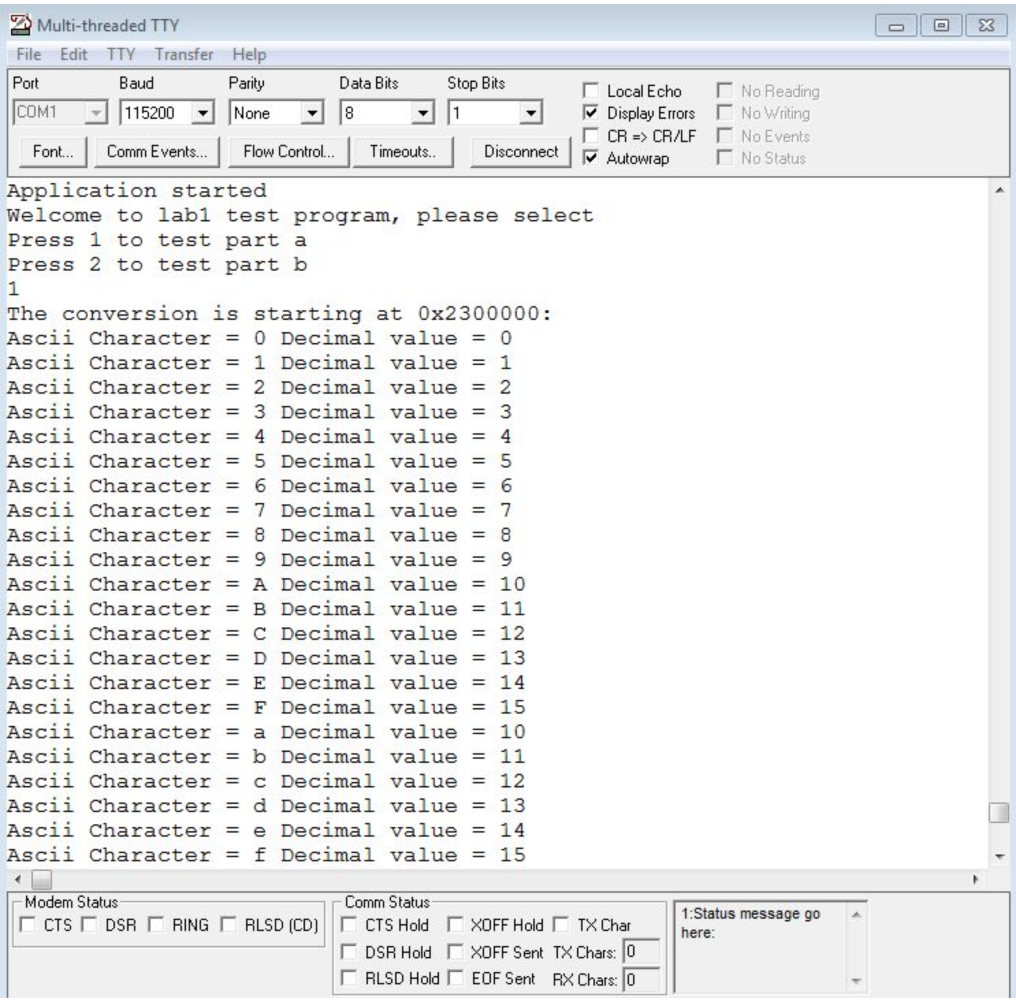
Sample calculation of conversion from uppercase to lowercase

eg. ASCII ' A ' or 0x41
- Uppercase, therefore add by 0x20 to convert to lowercase
  => 0x41 + 0x20 = 0x61
  => ASCII 'a'

Testing

Part A

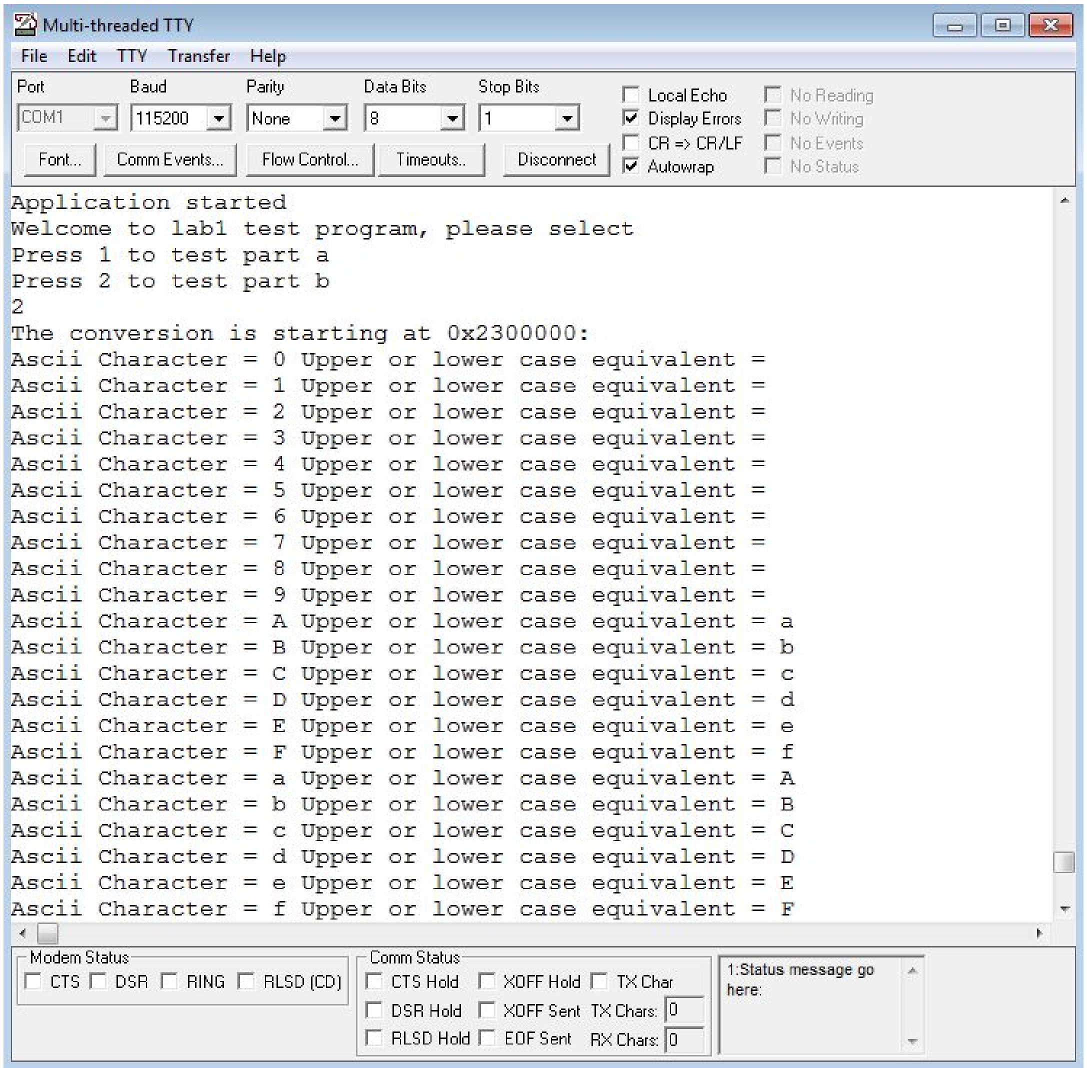
Inside the same directory, we have the files DataStorage.s, main.cpp, setZeros, Lab1Test.s from eclass. Lab1a.s was created by us. We then ran the program on MTTTY, and pressed 1. Since the values all matched up to the corresponding decimal values, our test was complete. The results can be shown in the picture down below.


Multi-threaded TTY
File Edit TTY Transfer Help
Port COM1
Baud 115200
Parity None
Data Bits 8
Stop Bits 1
Font... Comm Events... Flow Control... Timeouts.. Disconnect
□ Local Echo
☑ Display Errors
□ CR => CR/LF
☑ Autowrap
□ No Reading
□ No Writing
□ No Events
□ No Status
Modem Status □ CTS □ DSR □ RING □ RLSD (CD)
Comm Status □ CTS Hold □ XOFF Hold □ TX Char □ DSR Hold □ XOFF Sent TX Chars: 0 □ RLSD Hold □ EOF Sent RX Chars: 0
1:Status message go here:

```
Application started
Welcome to lab1 test program, please select
Press 1 to test part a
Press 2 to test part b
1
The conversion is starting at 0x2300000:
Ascii Character = 0 Decimal value = 0
Ascii Character = 1 Decimal value = 1
Ascii Character = 2 Decimal value = 2
Ascii Character = 3 Decimal value = 3
Ascii Character = 4 Decimal value = 4
Ascii Character = 5 Decimal value = 5
Ascii Character = 6 Decimal value = 6
Ascii Character = 7 Decimal value = 7
Ascii Character = 8 Decimal value = 8
Ascii Character = 9 Decimal value = 9
Ascii Character = A Decimal value = 10
Ascii Character = B Decimal value = 11
Ascii Character = C Decimal value = 12
Ascii Character = D Decimal value = 13
Ascii Character = E Decimal value = 14
Ascii Character = F Decimal value = 15
Ascii Character = a Decimal value = 10
Ascii Character = b Decimal value = 11
Ascii Character = c Decimal value = 12
Ascii Character = d Decimal value = 13
Ascii Character = e Decimal value = 14
Ascii Character = f Decimal value = 15
```

Part B

Inside the same directory, we have the files DataStorage.s, main.cpp, setZeros, Lab1Test.s from eclass. Lab1b.s was created by us. We then ran the program on MTTTY, and pressed 2. Since the values did not output when it wasn't a character, and the character swapped between upper and lower case, our test was complete. The results can be shown in the picture down below.



Multi-threaded TTY
File Edit TTY Transfer Help
Port COM1
Baud 115200
Parity None
Data Bits 8
Stop Bits 1
Font... Comm Events... Flow Control... Timeouts.. Disconnect
□ Local Echo
☑ Display Errors
□ CR => CR/LF
☑ Autowrap
□ No Reading
□ No Writing
□ No Events
□ No Status
Modem Status □ CTS □ DSR □ RING □ RLSD (CD)
Comm Status □ CTS Hold □ XOFF Hold □ TX Char □ DSR Hold □ XOFF Sent TX Chars: 0 □ RLSD Hold □ EOF Sent RX Chars: 0
1:Status message go here:

```
Application started
Welcome to lab1 test program, please select
Press 1 to test part a
Press 2 to test part b
2
The conversion is starting at 0x2300000:
Ascii Character = 0 Upper or lower case equivalent =
Ascii Character = 1 Upper or lower case equivalent =
Ascii Character = 2 Upper or lower case equivalent =
Ascii Character = 3 Upper or lower case equivalent =
Ascii Character = 4 Upper or lower case equivalent =
Ascii Character = 5 Upper or lower case equivalent =
Ascii Character = 6 Upper or lower case equivalent =
Ascii Character = 7 Upper or lower case equivalent =
Ascii Character = 8 Upper or lower case equivalent =
Ascii Character = 9 Upper or lower case equivalent =
Ascii Character = A Upper or lower case equivalent = a
Ascii Character = B Upper or lower case equivalent = b
Ascii Character = C Upper or lower case equivalent = c
Ascii Character = D Upper or lower case equivalent = d
Ascii Character = E Upper or lower case equivalent = e
Ascii Character = F Upper or lower case equivalent = f
Ascii Character = a Upper or lower case equivalent = A
Ascii Character = b Upper or lower case equivalent = B
Ascii Character = c Upper or lower case equivalent = C
Ascii Character = d Upper or lower case equivalent = D
Ascii Character = e Upper or lower case equivalent = E
Ascii Character = f Upper or lower case equivalent = F
```

Questions

1. *What happens when there is no exit code '0x0D' provided in the initialization process? Would it cause a problem? Why or why not?*

Answer: The program would run an infinite loop and would never go to the next step which is showing the value outputs, and thus will not terminal the program. In conclusion this will cause a problem.

2. *How can our code be modified to provide a variable address range? For example, what if I only wanted to convert the first 10 data entries?*

Answer: There are many ways to do it. One way to do it is to limit our memory address to being less than 0x43000028, which is 10 entries by comparing and doing an if is less statement (blt) for example. Another way is to use a counter starting at 0 counting 1 every time it loops inside the program, and stop looping when it counts to 10 by comparing and doing a if is less statement (blt) for example.

Conclusion

Part A of the lab required us to convert the ASCII characters 0-9, A-F, and a-f to their decimal value equivalent. Part B was to convert all the ASCII letters to their upper or lower case equivalent. If the program comes across an invalid ASCII character, the Error Code 0xFFFFFFFF would be stored in memory instead of the converted value. The Enter Code 0x0D signaled the end of the program and that there would be no more values to be converted. In order to carry out this lab, we used basic commands such as MOVE, CMPI, BEQ, ADD, and more in our assembly code. We then tested our codes by running them on MTTTY and ended up obtaining the expected results.

Appendix

Part A Assembler Code(lab1a.s)

```
/* DO NOT MODIFY THIS -------------------------------------------*/
.text

.global AssemblyProgram

AssemblyProgram:
lea      -40(%a7),%a7 /*Backing up data and address registers */
movem.l %d2-%d7/%a2-%a5,(%a7)
/*-------------------------------------------------------------*/

/*********************************************************************/
/* General Information *********************************************/
/* File Name: Lab1a.s **********************************************/
/* Names of Students: Darius Fang and Marlene Gong **************/
/* IDs: 1570975 and 1572719**************************************/
/* Date: Feb 5 2020                                                **/
/* General Description: Conversion of ASCII characters to decimal**/
/* values                                                          **/
/********************************************************************/

/*Write your program here*******************************************/


movea.l #0x43000000, %a2        /* set a2 to point at memory location 0x43000000 */
move.l (%a2), %d2               /* move contents of 0x43000000 to d2 */

movea.l #0x43100000, %a3        /* set a3 to point at memory location 0x43100000 */
move.l (%a3), %d3               /* move contents of 0x43100000 to d3 */

cmpi.l #0x0d, %d2               /* compare contents of d2 to the Enter code */
beq done                        /* if equal, exit program */
bra compare                     /* otherwise, proceed to 'compare' */

LOOP:
move.l %d3, (%a3)               /* move contents of d3 to where a3 is pointing */
adda.l #4, %a2                  /* set a2 to point at next memory location */
move.l (%a2), %d2               /* move contents of new memory location to d2 */

adda.l #4, %a3                  /* set a3 to point at next memory location */
move.l (%a3), %d3               /* move contents of new memory location to d3 */

cmpi.l #0x0d, %d2               /* compare contents of d2 to Enter code */
beq done                        /* if equal, exit program */
bra compare                     /* otherwise, proceed to 'compare' */

compare:
cmpi.l #0x30, %d2               /* compare contents of d2 to 0x30 */
blt ERROR                       /* if contents are less than, proceed to 'ERROR' */
```

```
cmpi.l #0x3A, %d2               /* compare contents of d2 to 0x3A (Decimal ASCII) */
blt DEC                         /* if contents are less than, proceed to 'DEC' */

cmpi.l #0x40, %d2               /* compare contents of d2 to 0x40 */
blt ERROR                       /* if contents are less than, proceed to 'ERROR' */

cmpi.l #0x47, %d2               /* compare contents of d2 to 0x47(A-F ASCII) */
blt UPPER                       /* if contents are less than, proceed to 'UPPER' */

cmpi.l #0x61, %d2               /* compare contents of d2 to 0x61 */
blt ERROR                       /* if contents are less than, proceed to 'ERROR' */

cmpi.l #0x67, %d2               /* compare contents of d2 to 0x67(a-f ASCII) */
blt LOWER                       /* if contents are less than, proceed to 'LOWER' */
bra ERROR                       /* otherwise, proceed to 'ERROR' */

ERROR:
move.l #0xFFFFFFFF, %d3         /* move Error Code to d3 */
bra LOOP                        /* proceed to 'LOOP' */

DEC:
move.l %d2, %d3                 /* move contents of d2 to d3 */
sub.l #0x30, %d3                /* subtract 0x30 from contents of d3 */
bra LOOP                        /* proceed to 'LOOP' */

LOWER:
move.l %d2, %d3                 /* move contents of d2 to d3 */
sub.l #87, %d3                  /* subtract 87(0x57) from contents of d3 */
bra LOOP                        /* proceed to 'LOOP' */

UPPER:
move.l %d2, %d3                 /* move contents of d2 to d3 */
sub.l #55, %d3                  /* subtract 55(0x39)  from contents of d3 */
bra LOOP                        /* proceed to 'LOOP' */

done:


/*End of program *************************************************/

/* DO NOT MODIFY THIS --------------------------------------------*/
movem.l (%a7),%d2-%d7/%a2-%a5 /*Restore data and address registers */
lea     40(%a7),%a7
rts
/*-----------------------------------------------------------------*/
```

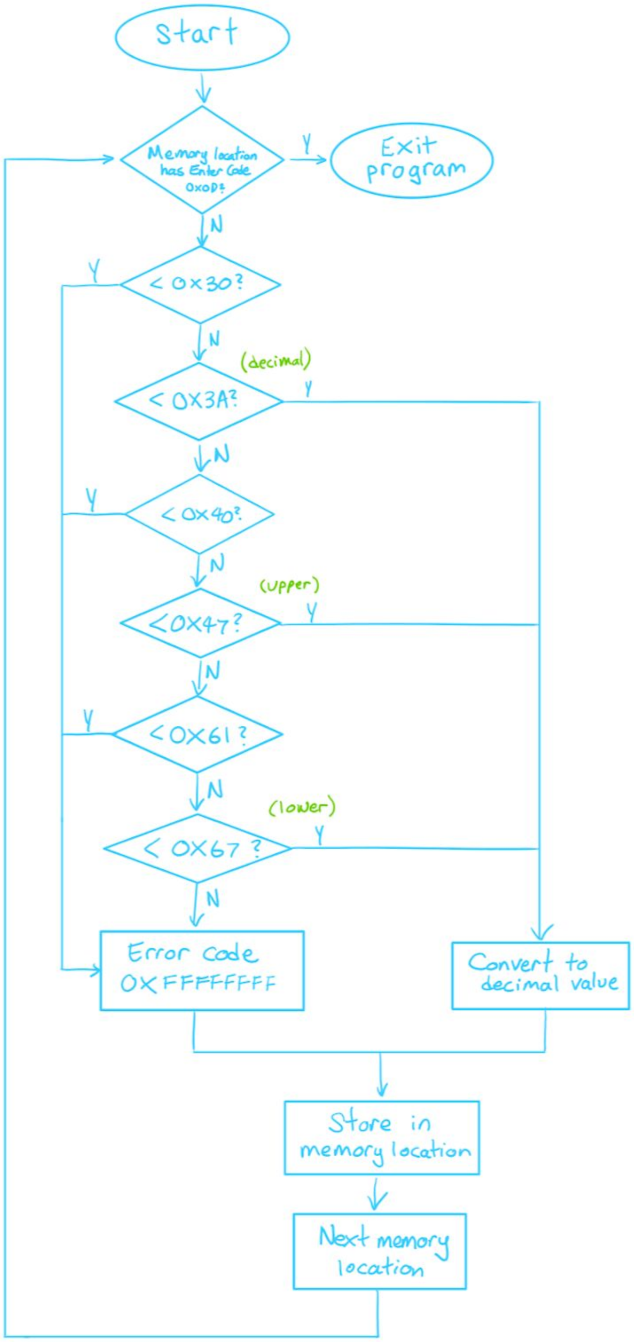
Part B Assembler Code(lab1b.s)

```
/* DO NOT MODIFY THIS ------------------------------------------*/
.text

.global AssemblyProgram

AssemblyProgram:
lea     -40(%a7),%a7 /*Backing up data and address registers */
movem.l %d2-%d7/%a2-%a5,(%a7)
/******************************************************************/
/* General Information ******************************************/
/* File Name: lab1b.s ********************************************/
/* Names of Students: Darius Fang and Marlene Gong**************/
/* IDs: 1570975 and 1572719*************************************/
/* Date: Feb 5 2020                                          **/
/* General Description: Conversion of uppercase letters to   **/
/* lowercase letters and vice versa                          **/
/****************************************************************/

/*Write your program here*****************************************/


movea.l #0x43000000, %a2        /* set a2 to point at memory location 0x43000000 */
move.l (%a2), %d2               /* move contents of 0x43000000 to d2 */

movea.l #0x43200000, %a3        /* set a3 to point at memory location 0x43200000 */
move.l (%a3), %d3               /* move contents of 0x43200000 to d3 */


cmpi.l #0x0d, %d2               /* compare contents of d2 to the Enter code */
beq done                        /* if equal, exit program */
bra compare                     /* otherwise, proceed to 'compare' */

LOOP:
adda.l #4, %a2                  /* set a2 to point at next memory location */
move.l (%a2), %d2               /* move contents of new memory location to d2 */

adda.l #4, %a3                  /* set a3 to point at next memory location */
move.l (%a3), %d3               /* move contents of new memory location to d3 */

cmpi.l #0x0d, %d2               /* compare contents of d2 to Enter code */
beq done                        /* if equal, exit program */
bra compare                     /* otherwise, proceed to 'compare' */



compare:
cmpi.l #0x41, %d2               /* compare contents of d2 to 0x41 */
blt ERROR                       /* if contents are less than, proceed to 'ERROR' */

cmpi.l #0x5B, %d2               /* compare contents of d2 to 0x5B (A-Z ASCII)*/
blt UPPER                       /* if contents are less than, proceed to 'UPPER' */
```

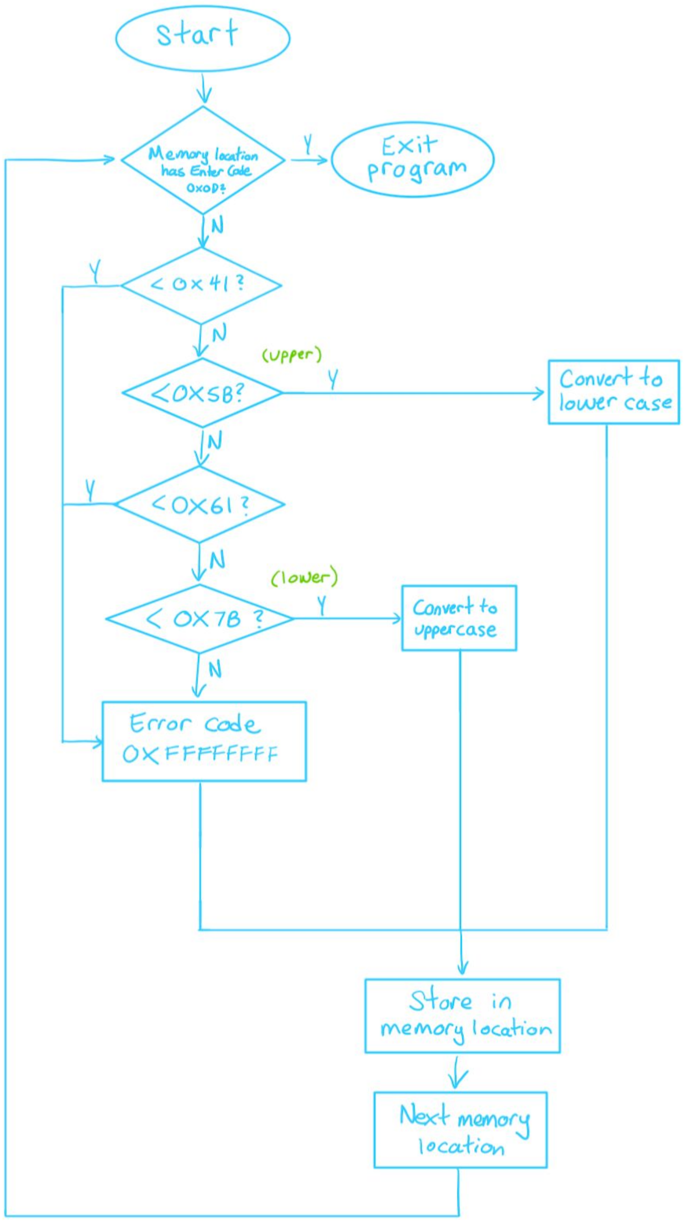
```
cmpi.l #0x61, %d2              /* compare contents of d2 to 0x61 */
blt ERROR                      /* if contents are less than, proceed to 'ERROR' */

cmpi.l #0x7B, %d2              /* compare contents of d2 to 0x7B(a-z) */
blt LOWER                      /* if contents are less than, proceed to 'LOWER' */
bra ERROR                      /* otherwise, proceed to 'ERROR' */

ERROR:
move.l #0xFFFFFFFF, %d3        /* move Error Code to d3 */
move.l %d3, (%a3)              /* move contents of d3 to where a3 is pointing */
bra LOOP                       /* proceed to 'LOOP' */

LOWER:
move.l %d2, %d3                /* move contents of d2 to d3 */
sub.l #0x20, %d3               /* subtract 0x20 from contents of d3 (to Lowercase ASCII)*/
move.l %d3, (%a3)              /* move contents of d3 to where a3 is pointing */
bra LOOP                       /* proceed to 'LOOP' */

UPPER:
move.l %d2, %d3                /* move contents of d2 to d3 */
add.l #0x20, %d3               /* add 0x20 to contents of d3(to Uppercase ASCII) */
move.l %d3, (%a3)              /* move contents of d3 to where a3 is pointing */
bra LOOP                       /* proceed to 'LOOP' */

done:

/*End of program *************************************************/

/* DO NOT MODIFY THIS --------------------------------------------*/
movem.l (%a7),%d2-%d7/%a2-%a5 /*Restore data and address registers */
lea     40(%a7),%a7
rts
/*----------------------------------------------------------------*/
```

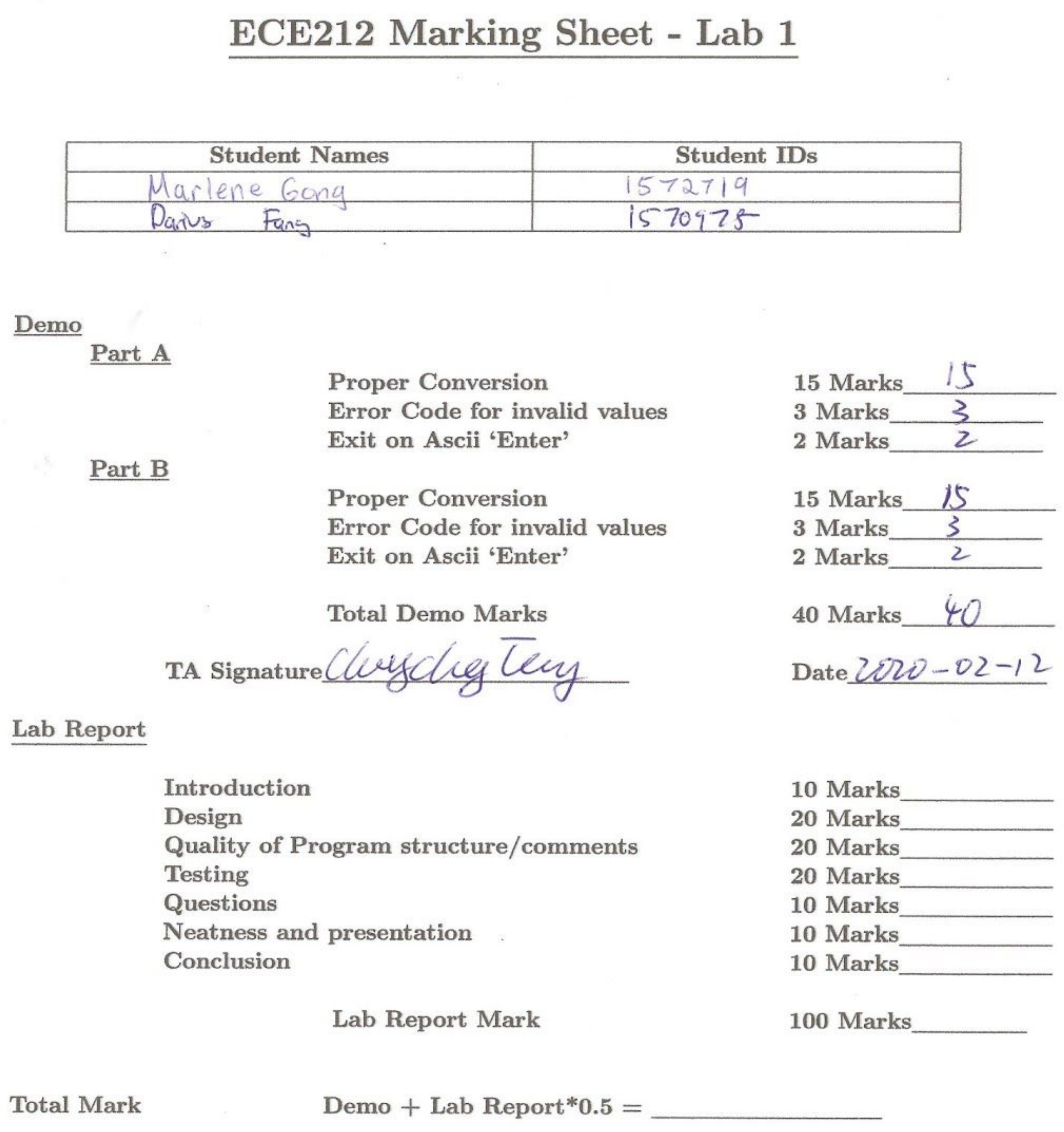
Flowcharts

Part A (lab1a.s)



Part B (lab1b.s)





ECE212 Marking Sheet - Lab 1

Student Names	Student IDs
Marlene Gong	1572719
Darius Fang	1570975

Demo

Part A

Proper Conversion	15 Marks	15
Error Code for invalid values	3 Marks	3
Exit on Ascii 'Enter'	2 Marks	2

Part B

Proper Conversion	15 Marks	15
Error Code for invalid values	3 Marks	3
Exit on Ascii 'Enter'	2 Marks	2
Total Demo Marks	40 Marks	40

TA Signature________ Date 2020-02-12

Lab Report

Introduction	10 Marks	
Design	20 Marks	
Quality of Program structure/comments	20 Marks	
Testing	20 Marks	
Questions	10 Marks	
Neatness and presentation	10 Marks	
Conclusion	10 Marks	
Lab Report Mark	100 Marks	

Total Mark Demo + Lab Report*0.5 = __________