# CIS11 Course Project Part 1: Documenting the Project

Fill in the following areas (purple).

**Introduction**

* 1. **Purpose**

**The Purpose of a bubble sort is to sort algorithm by repeatedly checks two elements, and swaps them if they are wrong. It is a simple way to sort data. Example is (4123)>(1423)>(1243)>(1234)**

* 1. **Intended Audience and Users**

Primary audience would be someone who is looking to sort a small amount of data.

* 1. **Product Scope**

The intention of this program is to sort/organize the elements/numbers in the order of smallest to biggest.

* 1. **Reference**

**Source Documents for the Program Requirements and Specification**

Reference Project requirements and LC-3 specifications.

Assembly Language pg x86 Processors

Irvine, Kip R. Assembly Language for x86 Processors. Pearson, 2015.

**Companion Application Requirements Documents (If applicable)**

1. Flow Chart
2. Psuedo Code
3. Program Code.
4. Cis Project Part 1 task assignment
5. Cis 11 Course Project part 1 documentation

**2. Overall Description**

**2.1 Product Perspective**

Primary program objectives

The primary program objective is a straight forward way to sorting algorithm. This way of sorting is by comparing the data next to each and seeing which element is bigger. Then repeats process until the order is right.

* 1. **Product Functions**

**The overall description of functionality:**

Highlight the program functionality: Identify tasks and subtasks of the program in summary.

Example:

1. Provide facilities to enhance the exchange of information among faculty and staff during curriculum development.  Do so by enabling distribution of official information with ancillary discussion among authorized faculty members, staff, and faculty committees during all phases of subject proposal development and review, including prior to proposal submission to the COC/CGSP.
2. Preserve a record of these decisions and their context.
3. Support versioning and workflow management of the information that it maintains.
4. Replace the current catalog production system, in which departments submit subject listing changes both electronically and on paper and curricular changes on paper, with a fully electronic system.  (However, printed listings will still be obtainable upon request.)
5. Enable updating of catalog data throughout the year. Do so for more than one term/year simultaneously.
6. Provide up-to-date information about subjects, schedules, and instructors to the MIT community (faculty, academic staff, students, alumni, and prospective students).
7. Provide easy-to-use, on-demand print and on-line publishing.  Non-subject data now printed in the MIT bulletin will be integrated via the web with subject data for integrated publishing.

**Technical functionality**

A configurable toolkit of functions including:

What are the technical functions of the program? Subroutines and operations.

Example:

        Ability to define new fields to capture for certain types of data (extensibility)

        Ability to configure fields, their sequencing, and formatting (i.e. style tags) for downloads so that any type of publication (print or web) can be downloaded without specialized programming.

        Flexible form generation including user-configurable field layout, text descriptors.

Reusable components for most functionality.

Use of Java and the new SSIT Internet platform, and when appropriate, XML

* 1. **User Classes and Characteristics**

**Who are involved in this development process? Include business and technical personnel and their tasks.**

**Example:**

**Academic Services personnel**

Responsible for the overall tracking and publishing of the MIT catalog.

Support the development of new and changed subject proposals.

Support COC and CGSP review of new and changed proposals.

Pre-register and register students. Manage Add and Drop requests.

Schedule classrooms, students, and finals.

Manage and report on pre-requisites, co-requisites.

Audit student degrees (GIR)

**Department Coordinators**

Responsible for helping faculty develop MIT catalog and related information for their department.

Monitor departmental roadmaps

Help develop room schedules for subjects and exams

Audit department degrees

**COC and CGSP**

Review subject proposals

**Other Administrative Offices**

The HASS Office, PSB, Communications Office review and support the development of the MIT catalog and supplemental bulletins.

Run student lotteries.

Submit grades.

**Faculty**

Plan and teach curricula

Use many reports provided by Academic Services: class lists, etc.

**Students**

Use catalog and related information to plan course work.

Use the on-line planning worksheet, lottery submittal, and pre-registration functions.

* 1. **Operating Environment**

The operating System will be using is Windows 10 and the development platform is github and LC-3

* 1. **Design and Implementation Constraints**

The limitation of this program is that the person needs to be familiar with using a LC3 Simulator program.

* 1. **Assumptions and Dependencies**

Assumptions for this program is that the person has a basic knowledge of computers. For an example like downloading, using a keyboard, a mouse, the internet, able to navigate the computer and able to understand how to use LC3 Simulator to run the program.

***3*. External Interface Requirements**

* 1. **User Interfaces**

The menu will be very simple to the point. On the prompt it will say “ “. Just follow what prompt says and it will be easier for the user to use. The user is able to click on the prompt and write down any number for what the prompt says.

* 1. **Hardware Interfaces**

Any computer can use this program if they can download LC3 Simulator.

* 1. **Software Interfaces**

The software needed to run the program is LC3 Simulator. The program can be giving to download.

* 1. Communications Interface

The program does not require any internet connection, but need an internet connection to download the simulator to run the program.

**4. Detailed Description of Functional requirements**

**4.1     Type of Requirement (summarize from Section 2.2)**

**Registration Form Requirement**

 Purpose: sorting algorithms

Input: By pressing the key/number pad to input our number.

 Processing: User inputs numbers into the prompt and what the program does is checks the numbers/elements by smallest to biggest by compare the first two and does this until next 8 numbers.

Outputs: The order of the number inputs will show in the next page by smallest to biggest. If an input is incorrect an error message will appear.

Data: User database

**Analysis Requirement**

Example:

Purpose: The research question is selected to perform analysis like regression.  
  
Inputs: Input will be the research question selected by the user and consequently the data that the user wants to use for the analysis.

Processing:  Depending on the research question, the appropriate statistical analysis is performed with the help of the EJB which provide the middle layer in this three tier application. It can be regression analysis, correlation, hypothesis test or the chi square test.

If an invalid input is entered, there will be appropriate error messages handled by using java exceptions in the java programs (EJB’s and Servlets). The HTML will include java scripts to handle error checks at the client side. Thus both client-side and server-side errors and exceptions will be handled completely within the application. The SAT will also undergo rigorous testing with various inputs to check whether the analysis is being conducted correctly and that all invalid inputs are not accepted.

Outputs: The output will be a graph or table of the analysis results displayed on the web browser page.

Data: Report database

**4.2 Performance requirements  
 What is the expected performance level of the program?**

**4.2.1** The application should be portable and possible to users of Netscape Navigator as well as Internet Explorer.

4.2.2 Since the application will be displaying graphs for the analysis, the response time for a particular analysis should be not be greater than 3-4 seconds for a respectable internet connection speed.

4.2.3 The database should be scalable; it must have the capacity to hold large number of users in future.

**4.2.4** Error handling should be implemented and the application should be able to handle all run time errors.

**4.3 Flow Chart and Pseudocode.**

|  |
| --- |
|  |
|  | http://web.mit.edu/ssit/cis/CISRequirements_files/image001.gif |

|  |
| --- |
|  |
|  | http://web.mit.edu/ssit/cis/CISRequirements_files/image002.gif |

**Pseudocode:**

.orig x3000

Get the user to input the 8 numbers to sort

{

First Number Input

{

Load NC to R6 ;Counts how many numbers are there to sort

LOAD Inverse\_ASCII\_OFFSET to R5

First Value ;First value can only be 1 or 0

{

LEA R0, NumInput1 ;Ask user to input the first value of a 3 digit number

PUTS ;display question

GETC ;ask for input

ADD R1, R5, R0 ;Copy input to R1 and place ascii offset

;Check if 1st value is 1, 0 or more than 1

ADD R2, R2, #1 ;Add 1 to R2

Two's complement R2

R1 minus R2 ;1-1 BRz or 0-1 BRn or 2-1 BRp

BRp Halt to error ;If 1st value higher than 1, error because 100 is max value.

BRn JMP to ask 2nd value input

BRz

Check if value is 100, if over 100, error

{

;If 1st value is 1, input should be 100. 2nd and 3rd should be zero anything else is error

;Second Value check

BRz

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue to ask third value, 1+0 if positive halt to error

BRpn Halt to error

;Third Value Check ;input should be 0 anything else is error

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue, 1+0 if positive halt to error

BRpn Error ;Jump to Halt

BRz

ADD R6, R6, #-1 ;If zero, remove 1 from Number Loop and ask for 2nd number to sort

Multiply Loop ;If input is 100, convert input to value 100 and store to number

{

Multloop ;Create value 100 by 10 x 10.

ADD R3, R3, #10

ADD R1, R1, #10

ADD R3, R3, #-1

BRp Multloop

Store Number to Num1

}

}

}

Second Value

{

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number, value can be 0-9

PUTS ;value is in the 10s so # times 10.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

BRz ADD R4, R1, #0 ;If 2nd value is zero, Copy R1 to R4 for later to add with 3rd value

JMP Third Value ;then jump to third value, if not continue to multloop

Multloop2

{

R1 is counter for mult loop ;IF 3, 3X10 FOR 30

ADD R2, R2, #10

ADD R1, R1, #-1

BRp MULTLOOP2

ADD R4, R2, #0 ;Copy R2 to R4 for later to add with 3rd value

}

}

Third Value

{

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number, value can be 0-9

PUTS ;value is in the 1s so just add with second value ex. 30 + 9 = 39.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R4, R1 ;ADD R4 to R1 Add the 10s value with the 1s value ex. 30 + 9 = 39.

Store R2 to Num1 ;Then continue to ask for next 3 digit number

}

}

Second Number Input

{

Load NC to R6 ;Counts how many numbers are there to sort

LOAD Inverse\_ASCII\_OFFSET to R5

First Value ;First value can only be 1 or 0

{

LEA R0, NumInput1 ;Ask user to input the first value of a 3 digit number

PUTS ;display question

GETC ;ask for input

ADD R1, R5, R0 ;Copy input to R1 and place ascii offset

;Check if 1st value is 1, 0 or more than 1

ADD R2, R2, #1 ;Add 1 to R2

Two's complement R2

R1 minus R2 ;1-1 BRz or 0-1 BRn or 2-1 BRp

BRp Halt to error ;If 1st value higher than 1, error because 100 is max value.

BRn JMP to ask 2nd value input

BRz

Check if value is 100, if over 100, error

{

;If 1st value is 1, input should be 100. 2nd and 3rd should be zero anything else is error

;Second Value check

BRz

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue to ask third value, 1+0 if positive halt to error

BRpn Halt to error

;Third Value Check ;input should be 0 anything else is error

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue, 1+0 if positive halt to error

BRpn Error ;Jump to Halt

BRz

ADD R6, R6, #-1 ;If zero, remove 1 from Number Loop and ask for 2nd number to sort

Multiply Loop ;If input is 100, convert input to value 100 and store to number

{

Multloop ;Create value 100 by 10 x 10.

ADD R3, R3, #10

ADD R1, R1, #10

ADD R3, R3, #-1

BRp Multloop

Store Number to Num2

}

}

}

Second Value

{

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number, value can be 0-9

PUTS ;value is in the 10s so # times 10.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

BRz ADD R4, R1, #0 ;If 2nd value is zero, Copy R1 to R4 for later to add with 3rd value

JMP Third Value ;then jump to third value, if not continue to multloop

Multloop2

{

R1 is counter for mult loop ;IF 3, 3X10 FOR 30

ADD R2, R2, #10

ADD R1, R1, #-1

BRp MULTLOOP2

ADD R4, R2, #0 ;Copy R2 to R4 for later to add with 3rd value

}

}

Third Value

{

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number, value can be 0-9

PUTS ;value is in the 1s so just add with second value ex. 30 + 9 = 39.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R4, R1 ;ADD R4 to R1 Add the 10s value with the 1s value ex. 30 + 9 = 39.

Store R2 to Num2 ;Then continue to ask for next 3 digit number

}

}

Third Number Input

{

Load NC to R6 ;Counts how many numbers are there to sort

LOAD Inverse\_ASCII\_OFFSET to R5

First Value ;First value can only be 1 or 0

{

LEA R0, NumInput1 ;Ask user to input the first value of a 3 digit number

PUTS ;display question

GETC ;ask for input

ADD R1, R5, R0 ;Copy input to R1 and place ascii offset

;Check if 1st value is 1, 0 or more than 1

ADD R2, R2, #1 ;Add 1 to R2

Two's complement R2

R1 minus R2 ;1-1 BRz or 0-1 BRn or 2-1 BRp

BRp Halt to error ;If 1st value higher than 1, error because 100 is max value.

BRn JMP to ask 2nd value input

BRz

Check if value is 100, if over 100, error

{

;If 1st value is 1, input should be 100. 2nd and 3rd should be zero anything else is error

;Second Value check

BRz

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue to ask third value, 1+0 if positive halt to error

BRpn Halt to error

;Third Value Check ;input should be 0 anything else is error

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue, 1+0 if positive halt to error

BRpn Error ;Jump to Halt

BRz

ADD R6, R6, #-1 ;If zero, remove 1 from Number Loop and ask for 2nd number to sort

Multiply Loop ;If input is 100, convert input to value 100 and store to number

{

Multloop ;Create value 100 by 10 x 10.

ADD R3, R3, #10

ADD R1, R1, #10

ADD R3, R3, #-1

BRp Multloop

Store Number to Num3

}

}

}

Second Value

{

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number, value can be 0-9

PUTS ;value is in the 10s so # times 10.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

BRz ADD R4, R1, #0 ;If 2nd value is zero, Copy R1 to R4 for later to add with 3rd value

JMP Third Value ;then jump to third value, if not continue to multloop

Multloop2

{

R1 is counter for mult loop ;IF 3, 3X10 FOR 30

ADD R2, R2, #10

ADD R1, R1, #-1

BRp MULTLOOP2

ADD R4, R2, #0 ;Copy R2 to R4 for later to add with 3rd value

}

}

Third Value

{

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number, value can be 0-9

PUTS ;value is in the 1s so just add with second value ex. 30 + 9 = 39.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R4, R1 ;ADD R4 to R1 Add the 10s value with the 1s value ex. 30 + 9 = 39.

Store R2 to Num3 ;Then continue to ask for next 3 digit number

}

}

Fourth Number Input

{

Load NC to R6 ;Counts how many numbers are there to sort

LOAD Inverse\_ASCII\_OFFSET to R5

First Value ;First value can only be 1 or 0

{

LEA R0, NumInput1 ;Ask user to input the first value of a 3 digit number

PUTS ;display question

GETC ;ask for input

ADD R1, R5, R0 ;Copy input to R1 and place ascii offset

;Check if 1st value is 1, 0 or more than 1

ADD R2, R2, #1 ;Add 1 to R2

Two's complement R2

R1 minus R2 ;1-1 BRz or 0-1 BRn or 2-1 BRp

BRp Halt to error ;If 1st value higher than 1, error because 100 is max value.

BRn JMP to ask 2nd value input

BRz

Check if value is 100, if over 100, error

{

;If 1st value is 1, input should be 100. 2nd and 3rd should be zero anything else is error

;Second Value check

BRz

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue to ask third value, 1+0 if positive halt to error

BRpn Halt to error

;Third Value Check ;input should be 0 anything else is error

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue, 1+0 if positive halt to error

BRpn Error ;Jump to Halt

BRz

ADD R6, R6, #-1 ;If zero, remove 1 from Number Loop and ask for 2nd number to sort

Multiply Loop ;If input is 100, convert input to value 100 and store to number

{

Multloop ;Create value 100 by 10 x 10.

ADD R3, R3, #10

ADD R1, R1, #10

ADD R3, R3, #-1

BRp Multloop

Store Number to Num4

}

}

}

Second Value

{

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number, value can be 0-9

PUTS ;value is in the 10s so # times 10.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

BRz ADD R4, R1, #0 ;If 2nd value is zero, Copy R1 to R4 for later to add with 3rd value

JMP Third Value ;then jump to third value, if not continue to multloop

Multloop2

{

R1 is counter for mult loop ;IF 3, 3X10 FOR 30

ADD R2, R2, #10

ADD R1, R1, #-1

BRp MULTLOOP2

ADD R4, R2, #0 ;Copy R2 to R4 for later to add with 3rd value

}

}

Third Value

{

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number, value can be 0-9

PUTS ;value is in the 1s so just add with second value ex. 30 + 9 = 39.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R4, R1 ;ADD R4 to R1 Add the 10s value with the 1s value ex. 30 + 9 = 39.

Store R2 to Num4 ;Then continue to ask for next 3 digit number

}

}

Fifth Number Input

{

Load NC to R6 ;Counts how many numbers are there to sort

LOAD Inverse\_ASCII\_OFFSET to R5

First Value ;First value can only be 1 or 0

{

LEA R0, NumInput1 ;Ask user to input the first value of a 3 digit number

PUTS ;display question

GETC ;ask for input

ADD R1, R5, R0 ;Copy input to R1 and place ascii offset

;Check if 1st value is 1, 0 or more than 1

ADD R2, R2, #1 ;Add 1 to R2

Two's complement R2

R1 minus R2 ;1-1 BRz or 0-1 BRn or 2-1 BRp

BRp Halt to error ;If 1st value higher than 1, error because 100 is max value.

BRn JMP to ask 2nd value input

BRz

Check if value is 100, if over 100, error

{

;If 1st value is 1, input should be 100. 2nd and 3rd should be zero anything else is error

;Second Value check

BRz

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue to ask third value, 1+0 if positive halt to error

BRpn Halt to error

;Third Value Check ;input should be 0 anything else is error

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue, 1+0 if positive halt to error

BRpn Error ;Jump to Halt

BRz

ADD R6, R6, #-1 ;If zero, remove 1 from Number Loop and ask for 2nd number to sort

Multiply Loop ;If input is 100, convert input to value 100 and store to number

{

Multloop ;Create value 100 by 10 x 10.

ADD R3, R3, #10

ADD R1, R1, #10

ADD R3, R3, #-1

BRp Multloop

Store Number to Num5

}

}

}

Second Value

{

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number, value can be 0-9

PUTS ;value is in the 10s so # times 10.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

BRz ADD R4, R1, #0 ;If 2nd value is zero, Copy R1 to R4 for later to add with 3rd value

JMP Third Value ;then jump to third value, if not continue to multloop

Multloop2

{

R1 is counter for mult loop ;IF 3, 3X10 FOR 30

ADD R2, R2, #10

ADD R1, R1, #-1

BRp MULTLOOP2

ADD R4, R2, #0 ;Copy R2 to R4 for later to add with 3rd value

}

}

Third Value

{

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number, value can be 0-9

PUTS ;value is in the 1s so just add with second value ex. 30 + 9 = 39.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R4, R1 ;ADD R4 to R1 Add the 10s value with the 1s value ex. 30 + 9 = 39.

Store R2 to Num5 ;Then continue to ask for next 3 digit number

}

}

Sixth Number Input

{

Load NC to R6 ;Counts how many numbers are there to sort

LOAD Inverse\_ASCII\_OFFSET to R5

First Value ;First value can only be 1 or 0

{

LEA R0, NumInput1 ;Ask user to input the first value of a 3 digit number

PUTS ;display question

GETC ;ask for input

ADD R1, R5, R0 ;Copy input to R1 and place ascii offset

;Check if 1st value is 1, 0 or more than 1

ADD R2, R2, #1 ;Add 1 to R2

Two's complement R2

R1 minus R2 ;1-1 BRz or 0-1 BRn or 2-1 BRp

BRp Halt to error ;If 1st value higher than 1, error because 100 is max value.

BRn JMP to ask 2nd value input

BRz

Check if value is 100, if over 100, error

{

;If 1st value is 1, input should be 100. 2nd and 3rd should be zero anything else is error

;Second Value check

BRz

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue to ask third value, 1+0 if positive halt to error

BRpn Halt to error

;Third Value Check ;input should be 0 anything else is error

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue, 1+0 if positive halt to error

BRpn Error ;Jump to Halt

BRz

ADD R6, R6, #-1 ;If zero, remove 1 from Number Loop and ask for 2nd number to sort

Multiply Loop ;If input is 100, convert input to value 100 and store to number

{

Multloop ;Create value 100 by 10 x 10.

ADD R3, R3, #10

ADD R1, R1, #10

ADD R3, R3, #-1

BRp Multloop

Store Number to Num6

}

}

}

Second Value

{

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number, value can be 0-9

PUTS ;value is in the 10s so # times 10.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

BRz ADD R4, R1, #0 ;If 2nd value is zero, Copy R1 to R4 for later to add with 3rd value

JMP Third Value ;then jump to third value, if not continue to multloop

Multloop2

{

R1 is counter for mult loop ;IF 3, 3X10 FOR 30

ADD R2, R2, #10

ADD R1, R1, #-1

BRp MULTLOOP2

ADD R4, R2, #0 ;Copy R2 to R4 for later to add with 3rd value

}

}

Third Value

{

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number, value can be 0-9

PUTS ;value is in the 1s so just add with second value ex. 30 + 9 = 39.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R4, R1 ;ADD R4 to R1 Add the 10s value with the 1s value ex. 30 + 9 = 39.

Store R2 to Num6 ;Then continue to ask for next 3 digit number

}

}

Seventh Number Input

{

Load NC to R6 ;Counts how many numbers are there to sort

LOAD Inverse\_ASCII\_OFFSET to R5

First Value ;First value can only be 1 or 0

{

LEA R0, NumInput1 ;Ask user to input the first value of a 3 digit number

PUTS ;display question

GETC ;ask for input

ADD R1, R5, R0 ;Copy input to R1 and place ascii offset

;Check if 1st value is 1, 0 or more than 1

ADD R2, R2, #1 ;Add 1 to R2

Two's complement R2

R1 minus R2 ;1-1 BRz or 0-1 BRn or 2-1 BRp

BRp Halt to error ;If 1st value higher than 1, error because 100 is max value.

BRn JMP to ask 2nd value input

BRz

Check if value is 100, if over 100, error

{

;If 1st value is 1, input should be 100. 2nd and 3rd should be zero anything else is error

;Second Value check

BRz

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue to ask third value, 1+0 if positive halt to error

BRpn Halt to error

;Third Value Check ;input should be 0 anything else is error

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue, 1+0 if positive halt to error

BRpn Error ;Jump to Halt

BRz

ADD R6, R6, #-1 ;If zero, remove 1 from Number Loop and ask for 2nd number to sort

Multiply Loop ;If input is 100, convert input to value 100 and store to number

{

Multloop ;Create value 100 by 10 x 10.

ADD R3, R3, #10

ADD R1, R1, #10

ADD R3, R3, #-1

BRp Multloop

Store Number to Num7

}

}

}

Second Value

{

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number, value can be 0-9

PUTS ;value is in the 10s so # times 10.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

BRz ADD R4, R1, #0 ;If 2nd value is zero, Copy R1 to R4 for later to add with 3rd value

JMP Third Value ;then jump to third value, if not continue to multloop

Multloop2

{

R1 is counter for mult loop ;IF 3, 3X10 FOR 30

ADD R2, R2, #10

ADD R1, R1, #-1

BRp MULTLOOP2

ADD R4, R2, #0 ;Copy R2 to R4 for later to add with 3rd value

}

}

Third Value

{

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number, value can be 0-9

PUTS ;value is in the 1s so just add with second value ex. 30 + 9 = 39.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R4, R1 ;ADD R4 to R1 Add the 10s value with the 1s value ex. 30 + 9 = 39.

Store R2 to Num7 ;Then continue to ask for next 3 digit number

}

}

Eight Number Input

{

Load NC to R6 ;Counts how many numbers are there to sort

LOAD Inverse\_ASCII\_OFFSET to R5

First Value ;First value can only be 1 or 0

{

LEA R0, NumInput1 ;Ask user to input the first value of a 3 digit number

PUTS ;display question

GETC ;ask for input

ADD R1, R5, R0 ;Copy input to R1 and place ascii offset

;Check if 1st value is 1, 0 or more than 1

ADD R2, R2, #1 ;Add 1 to R2

Two's complement R2

R1 minus R2 ;1-1 BRz or 0-1 BRn or 2-1 BRp

BRp Halt to error ;If 1st value higher than 1, error because 100 is max value.

BRn JMP to ask 2nd value input

BRz

Check if value is 100, if over 100, error

{

;If 1st value is 1, input should be 100. 2nd and 3rd should be zero anything else is error

;Second Value check

BRz

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue to ask third value, 1+0 if positive halt to error

BRpn Halt to error

;Third Value Check ;input should be 0 anything else is error

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number

PUTS

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R2, #0 ;Put 0 in R2

R1 plus R2 ;0+0 Continue, 1+0 if positive halt to error

BRpn Error ;Jump to Halt

BRz

ADD R6, R6, #-1 ;If zero, remove 1 from Number Loop and ask for 2nd number to sort

Multiply Loop ;If input is 100, convert input to value 100 and store to number

{

Multloop ;Create value 100 by 10 x 10.

ADD R3, R3, #10

ADD R1, R1, #10

ADD R3, R3, #-1

BRp Multloop

Store Number to Num8

}

}

}

Second Value

{

LEA R0, NumInput2 ;Ask user to input the second value of a 3 digit number, value can be 0-9

PUTS ;value is in the 10s so # times 10.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

BRz ADD R4, R1, #0 ;If 2nd value is zero, Copy R1 to R4 for later to add with 3rd value

JMP Third Value ;then jump to third value, if not continue to multloop

Multloop2

{

R1 is counter for mult loop ;IF 3, 3X10 FOR 30

ADD R2, R2, #10

ADD R1, R1, #-1

BRp MULTLOOP2

ADD R4, R2, #0 ;Copy R2 to R4 for later to add with 3rd value

}

}

Third Value

{

LEA R0, NumInput3 ;Ask user to input the third value of a 3 digit number, value can be 0-9

PUTS ;value is in the 1s so just add with second value ex. 30 + 9 = 39.

GETC

ADD R1, R0, #0 ;Copy R0 to R1

ADD R2, R4, R1 ;ADD R4 to R1 Add the 10s value with the 1s value ex. 30 + 9 = 39.

Store R2 to Num8 ;Then continue to ask for next 3 digit number

}

}

}

Subroutine loop to check for the highest number

{

Sortloop

Load LC to R7

{ ;Compare Num1 & Num2

Load Num1 to R1, Num2 to R2

Make a copy of LC to R6,

R1 to R3,

R2 to R4

Two's complement R4

Subtract Num1 R3 to Num2 R4

;if Num1 is greater than Num2 swap positions and add 1 to LC in R6

{

Store Num1 R1 to Label Num2

Store Num2 R2 to Label Num1

ADD R6, R6, #1

}

Clear R1-R4

}

{ ;Compare Num2 & Num3

Load Num2 to R1, Num3 to R2

Make a copy of R1 to R3,

R2 to R4

Two's complement R4

Subtract Num2 R3 to Num3 R4

;if Num2 is greater than Num3 swap positions and add 1 to LC in R6

{

Store Num2 R1 to Label Num3

Store Num3 R2 to Label Num2

ADD R6, R6, #1

}

Clear R1-R4

}

{ ;Compare Num3 & Num4

Load Num3 to R1, Num4 to R2

Make a copy of R1 to R3,

R2 to R4

Two's complement R4

Subtract Num3 R3 to Num4 R4

;if Num3 is greater than Num4swap positions and add 1 to LC in R6

{

Store Num3 R1 to Label Num4

Store Num4 R2 to Label Num3

ADD R6, R6, #1

}

Clear R1-R4

}

{ ;Compare Num4 & Num5

Load Num4 to R1, Num5 to R2

Make a copy of R1 to R3,

R2 to R4

Two's complement R4

Subtract Num4 R3 to Num5 R4

;if Num4 is greater than Num5 swap positions and add 1 to LC in R6

{

Store Num4 R1 to Label Num5

Store Num5 R2 to Label Num4

ADD R6, R6, #1

}

Clear R1-R4

}

{ ;Compare Num5 & Num6

Load Num5 to R1, Num6 to R2

Make a copy of R1 to R3,

R2 to R4

Two's complement R4

Subtract Num5 R3 to Num6 R4

;if Num5 is greater than Num6 swap positions and add 1 to LC in R6

{

Store Num5 R1 to Label Num6

Store Num6 R2 to Label Num5

ADD R6, R6, #1

}

Clear R1-R4

}

{ ;Compare Num6 & Num7

Load Num6 to R1, Num7 to R2

Make a copy of R1 to R3,

R2 to R4

Two's complement R4

Subtract Num6 R3 to Num7 R4

;if Num6 is greater than Num7 swap positions and add 1 to LC in R6

{

Store Num6 R1 to Label Num7

Store Num7 R2 to Label Num6

ADD R6, R6, #1

}

Clear R1-R4

}

{ ;Compare Num7 & Num8

Load Num7 to R1, Num8 to R2

Make a copy of R1 to R3,

R2 to R4

Two's complement R4

Subtract Num7 R3 to Num8 R4

;if Num7 is greater than Num8 swap positions and add 1 to LC in R6

{

Store Num7 R1 to Label Num8

Store Num8 R2 to Label Num7

ADD R6, R6, #1

}

}

Copy LC R6 to R5

Store R5 to LC as the new loop counter

Subtract LC in R7 and R6 ;Everytime two numbers are swapped adds 1 counter to LC. if LC R7 minus LC R6 is not zero , loop because sorting is not done yet.

Loop to SortLoop if not equal to zero or negative

If sorted, continue or JSR to Display

}

Display the sorted numbers

{

Display First value

{

Load Num1 to R1

Load ASCII\_OFFSET to R6

Load DC to R5

Check if value is 100, if 0 skip to second value check

{

ADD R2, R2, #-100

ADD R2, R1, R2 ;100-100=0 BRz, 39-100 BRn

BRn

Clear R0 ;If 1st value is zero, print out first value as zero

ADD R0, R0, #0

OUT ;Then jump to 2nd value check

2nd value check ;If value is below 100, enter second value check

BRz ADD R0, R0, #1 ;If BRz, value is 100 so output 100

ADD R0, R1, R0 ;Puts 1 to R0 for output

OUT ;Prints 1 to console

ADD R0, R0, #-1 ;1-1 = 0

OUT ;Then print 0 twice

OUT ;Outputs value "100"

}

Second Value Check

Check if second value is below 10

ADD R2, R1, #-10 ;If value less than 10, then second value is 0 then third value is a single digit value

BRpz DivideLoop ;If more than or equal 10 go to DivideLoop

BRn ;find the 1s value

Clear R0 ;If 2nd value is zero, print out second value as zero

ADD R0, R0, #0

OUT

ADD R0, R6, R1 ;ASCII\_OFFSET the single digit value, store in R0 for display

OUT ;Display Single digit value

DivideLoop ;Divide by ten to find the 10s value

{

ADD R5, R5, #1 ;Add 1 to Division Counter

ADD R4, R1, #0 ;Copy R1 to R4

ADD R4, R4, #-10 ;51/10 is equal to 51- 10(5 times) with remainder 1

BRp DivideLoop ;If still positive, continue divide loop

}

BRn ;If division results in negative, 51-10(6 times) = -11 so

ADD R5, R5, #-1 ;Remove 1 from Divide Counter

ADD R4, R4, #10 ;Place back 10 51-10(5times) = 1

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0 R5= (5 times) so 50s

OUT ;Display second value on screen

ADD R0, R4, R6 ;ASCII\_OFFSET the remainder 1 and store to R0

OUT ;Output the remainder 1 as the third value.

BRz ;If exactly zero, then third value is zero 50 -10 (5 times) = 0

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0

OUT ;Display second value on screen

Clear R0

ADD R0,R0, #0

OUT ;Output last digit as zero

;After displaying all three values, continue to display Num2

}

}

Display Second value

{

Load Num2 to R1

Load ASCII\_OFFSET to R6

Load DC to R5

Check if value is 100, if 0 skip to second value check

{

ADD R2, R2, #-100

ADD R2, R1, R2 ;100-100=0 BRz, 39-100 BRn

BRn

Clear R0 ;If 1st value is zero, print out first value as zero

ADD R0, R0, #0

OUT ;Then jump to 2nd value check

2nd value check ;If value is below 100, enter second value check

BRz ADD R0, R0, #1 ;If BRz, value is 100 so output 100

ADD R0, R1, R0 ;Puts 1 to R0 for output

OUT ;Prints 1 to console

ADD R0, R0, #-1 ;1-1 = 0

OUT ;Then print 0 twice

OUT ;Outputs value "100"

}

Second Value Check

Check if second value is below 10

ADD R2, R1, #-10 ;If value less than 10, then second value is 0 then third value is a single digit value

BRpz DivideLoop ;If more than or equal 10 go to DivideLoop

BRn ;find the 1s value

Clear R0 ;If 2nd value is zero, print out second value as zero

ADD R0, R0, #0

OUT

ADD R0, R6, R1 ;ASCII\_OFFSET the single digit value, store in R0 for display

OUT ;Display Single digit value

DivideLoop ;Divide by ten to find the 10s value

{

ADD R5, R5, #1 ;Add 1 to Division Counter

ADD R4, R1, #0 ;Copy R1 to R4

ADD R4, R4, #-10 ;51/10 is equal to 51- 10(5 times) with remainder 1

BRp DivideLoop ;If still positive, continue divide loop

}

BRn ;If division results in negative, 51-10(6 times) = -11 so

ADD R5, R5, #-1 ;Remove 1 from Divide Counter

ADD R4, R4, #10 ;Place back 10 51-10(5times) = 1

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0 R5= (5 times) so 50s

OUT ;Display second value on screen

ADD R0, R4, R6 ;ASCII\_OFFSET the remainder 1 and store to R0

OUT ;Output the remainder 1 as the third value.

BRz ;If exactly zero, then third value is zero 50 -10 (5 times) = 0

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0

OUT ;Display second value on screen

Clear R0

ADD R0,R0, #0

OUT ;Output last digit as zero

;After displaying all three values, continue to display Num3

}

}

Display Third value

{

Load Num3 to R1

Load ASCII\_OFFSET to R6

Load DC to R5

Check if value is 100, if 0 skip to second value check

{

ADD R2, R2, #-100

ADD R2, R1, R2 ;100-100=0 BRz, 39-100 BRn

BRn

Clear R0 ;If 1st value is zero, print out first value as zero

ADD R0, R0, #0

OUT ;Then jump to 2nd value check

2nd value check ;If value is below 100, enter second value check

BRz ADD R0, R0, #1 ;If BRz, value is 100 so output 100

ADD R0, R1, R0 ;Puts 1 to R0 for output

OUT ;Prints 1 to console

ADD R0, R0, #-1 ;1-1 = 0

OUT ;Then print 0 twice

OUT ;Outputs value "100"

}

Second Value Check

Check if second value is below 10

ADD R2, R1, #-10 ;If value less than 10, then second value is 0 then third value is a single digit value

BRpz DivideLoop ;If more than or equal 10 go to DivideLoop

BRn ;find the 1s value

Clear R0 ;If 2nd value is zero, print out second value as zero

ADD R0, R0, #0

OUT

ADD R0, R6, R1 ;ASCII\_OFFSET the single digit value, store in R0 for display

OUT ;Display Single digit value

DivideLoop ;Divide by ten to find the 10s value

{

ADD R5, R5, #1 ;Add 1 to Division Counter

ADD R4, R1, #0 ;Copy R1 to R4

ADD R4, R4, #-10 ;51/10 is equal to 51- 10(5 times) with remainder 1

BRp DivideLoop ;If still positive, continue divide loop

}

BRn ;If division results in negative, 51-10(6 times) = -11 so

ADD R5, R5, #-1 ;Remove 1 from Divide Counter

ADD R4, R4, #10 ;Place back 10 51-10(5times) = 1

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0 R5= (5 times) so 50s

OUT ;Display second value on screen

ADD R0, R4, R6 ;ASCII\_OFFSET the remainder 1 and store to R0

OUT ;Output the remainder 1 as the third value.

BRz ;If exactly zero, then third value is zero 50 -10 (5 times) = 0

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0

OUT ;Display second value on screen

Clear R0

ADD R0,R0, #0

OUT ;Output last digit as zero

;After displaying all three values, continue to display Num4

}

}

Display Fourth value

{

Load Num4 to R1

Load ASCII\_OFFSET to R6

Load DC to R5

Check if value is 100, if 0 skip to second value check

{

ADD R2, R2, #-100

ADD R2, R1, R2 ;100-100=0 BRz, 39-100 BRn

BRn

Clear R0 ;If 1st value is zero, print out first value as zero

ADD R0, R0, #0

OUT ;Then jump to 2nd value check

2nd value check ;If value is below 100, enter second value check

BRz ADD R0, R0, #1 ;If BRz, value is 100 so output 100

ADD R0, R1, R0 ;Puts 1 to R0 for output

OUT ;Prints 1 to console

ADD R0, R0, #-1 ;1-1 = 0

OUT ;Then print 0 twice

OUT ;Outputs value "100"

}

Second Value Check

Check if second value is below 10

ADD R2, R1, #-10 ;If value less than 10, then second value is 0 then third value is a single digit value

BRpz DivideLoop ;If more than or equal 10 go to DivideLoop

BRn ;find the 1s value

Clear R0 ;If 2nd value is zero, print out second value as zero

ADD R0, R0, #0

OUT

ADD R0, R6, R1 ;ASCII\_OFFSET the single digit value, store in R0 for display

OUT ;Display Single digit value

DivideLoop ;Divide by ten to find the 10s value

{

ADD R5, R5, #1 ;Add 1 to Division Counter

ADD R4, R1, #0 ;Copy R1 to R4

ADD R4, R4, #-10 ;51/10 is equal to 51- 10(5 times) with remainder 1

BRp DivideLoop ;If still positive, continue divide loop

}

BRn ;If division results in negative, 51-10(6 times) = -11 so

ADD R5, R5, #-1 ;Remove 1 from Divide Counter

ADD R4, R4, #10 ;Place back 10 51-10(5times) = 1

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0 R5= (5 times) so 50s

OUT ;Display second value on screen

ADD R0, R4, R6 ;ASCII\_OFFSET the remainder 1 and store to R0

OUT ;Output the remainder 1 as the third value.

BRz ;If exactly zero, then third value is zero 50 -10 (5 times) = 0

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0

OUT ;Display second value on screen

Clear R0

ADD R0,R0, #0

OUT ;Output last digit as zero

;After displaying all three values, continue to display Num5

}

}

Display Fifth value

{

Load Num5 to R1

Load ASCII\_OFFSET to R6

Load DC to R5

Check if value is 100, if 0 skip to second value check

{

ADD R2, R2, #-100

ADD R2, R1, R2 ;100-100=0 BRz, 39-100 BRn

BRn

Clear R0 ;If 1st value is zero, print out first value as zero

ADD R0, R0, #0

OUT ;Then jump to 2nd value check

2nd value check ;If value is below 100, enter second value check

BRz ADD R0, R0, #1 ;If BRz, value is 100 so output 100

ADD R0, R1, R0 ;Puts 1 to R0 for output

OUT ;Prints 1 to console

ADD R0, R0, #-1 ;1-1 = 0

OUT ;Then print 0 twice

OUT ;Outputs value "100"

}

Second Value Check

Check if second value is below 10

ADD R2, R1, #-10 ;If value less than 10, then second value is 0 then third value is a single digit value

BRpz DivideLoop ;If more than or equal 10 go to DivideLoop

BRn ;find the 1s value

Clear R0 ;If 2nd value is zero, print out second value as zero

ADD R0, R0, #0

OUT

ADD R0, R6, R1 ;ASCII\_OFFSET the single digit value, store in R0 for display

OUT ;Display Single digit value

DivideLoop ;Divide by ten to find the 10s value

{

ADD R5, R5, #1 ;Add 1 to Division Counter

ADD R4, R1, #0 ;Copy R1 to R4

ADD R4, R4, #-10 ;51/10 is equal to 51- 10(5 times) with remainder 1

BRp DivideLoop ;If still positive, continue divide loop

}

BRn ;If division results in negative, 51-10(6 times) = -11 so

ADD R5, R5, #-1 ;Remove 1 from Divide Counter

ADD R4, R4, #10 ;Place back 10 51-10(5times) = 1

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0 R5= (5 times) so 50s

OUT ;Display second value on screen

ADD R0, R4, R6 ;ASCII\_OFFSET the remainder 1 and store to R0

OUT ;Output the remainder 1 as the third value.

BRz ;If exactly zero, then third value is zero 50 -10 (5 times) = 0

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0

OUT ;Display second value on screen

Clear R0

ADD R0,R0, #0

OUT ;Output last digit as zero

;After displaying all three values, continue to display Num6

}

}

Display Sixth value

{

Load Num6 to R1

Load ASCII\_OFFSET to R6

Load DC to R5

Check if value is 100, if 0 skip to second value check

{

ADD R2, R2, #-100

ADD R2, R1, R2 ;100-100=0 BRz, 39-100 BRn

BRn

Clear R0 ;If 1st value is zero, print out first value as zero

ADD R0, R0, #0

OUT ;Then jump to 2nd value check

2nd value check ;If value is below 100, enter second value check

BRz ADD R0, R0, #1 ;If BRz, value is 100 so output 100

ADD R0, R1, R0 ;Puts 1 to R0 for output

OUT ;Prints 1 to console

ADD R0, R0, #-1 ;1-1 = 0

OUT ;Then print 0 twice

OUT ;Outputs value "100"

}

Second Value Check

Check if second value is below 10

ADD R2, R1, #-10 ;If value less than 10, then second value is 0 then third value is a single digit value

BRpz DivideLoop ;If more than or equal 10 go to DivideLoop

BRn ;find the 1s value

Clear R0 ;If 2nd value is zero, print out second value as zero

ADD R0, R0, #0

OUT

ADD R0, R6, R1 ;ASCII\_OFFSET the single digit value, store in R0 for display

OUT ;Display Single digit value

DivideLoop ;Divide by ten to find the 10s value

{

ADD R5, R5, #1 ;Add 1 to Division Counter

ADD R4, R1, #0 ;Copy R1 to R4

ADD R4, R4, #-10 ;51/10 is equal to 51- 10(5 times) with remainder 1

BRp DivideLoop ;If still positive, continue divide loop

}

BRn ;If division results in negative, 51-10(6 times) = -11 so

ADD R5, R5, #-1 ;Remove 1 from Divide Counter

ADD R4, R4, #10 ;Place back 10 51-10(5times) = 1

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0 R5= (5 times) so 50s

OUT ;Display second value on screen

ADD R0, R4, R6 ;ASCII\_OFFSET the remainder 1 and store to R0

OUT ;Output the remainder 1 as the third value.

BRz ;If exactly zero, then third value is zero 50 -10 (5 times) = 0

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0

OUT ;Display second value on screen

Clear R0

ADD R0,R0, #0

OUT ;Output last digit as zero

;After displaying all three values, continue to display Num7

}

}

Display Seventh value

{

Load Num7 to R1

Load ASCII\_OFFSET to R6

Load DC to R5

Check if value is 100, if 0 skip to second value check

{

ADD R2, R2, #-100

ADD R2, R1, R2 ;100-100=0 BRz, 39-100 BRn

BRn

Clear R0 ;If 1st value is zero, print out first value as zero

ADD R0, R0, #0

OUT ;Then jump to 2nd value check

2nd value check ;If value is below 100, enter second value check

BRz ADD R0, R0, #1 ;If BRz, value is 100 so output 100

ADD R0, R1, R0 ;Puts 1 to R0 for output

OUT ;Prints 1 to console

ADD R0, R0, #-1 ;1-1 = 0

OUT ;Then print 0 twice

OUT ;Outputs value "100"

}

Second Value Check

Check if second value is below 10

ADD R2, R1, #-10 ;If value less than 10, then second value is 0 then third value is a single digit value

BRpz DivideLoop ;If more than or equal 10 go to DivideLoop

BRn ;find the 1s value

Clear R0 ;If 2nd value is zero, print out second value as zero

ADD R0, R0, #0

OUT

ADD R0, R6, R1 ;ASCII\_OFFSET the single digit value, store in R0 for display

OUT ;Display Single digit value

DivideLoop ;Divide by ten to find the 10s value

{

ADD R5, R5, #1 ;Add 1 to Division Counter

ADD R4, R1, #0 ;Copy R1 to R4

ADD R4, R4, #-10 ;51/10 is equal to 51- 10(5 times) with remainder 1

BRp DivideLoop ;If still positive, continue divide loop

}

BRn ;If division results in negative, 51-10(6 times) = -11 so

ADD R5, R5, #-1 ;Remove 1 from Divide Counter

ADD R4, R4, #10 ;Place back 10 51-10(5times) = 1

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0 R5= (5 times) so 50s

OUT ;Display second value on screen

ADD R0, R4, R6 ;ASCII\_OFFSET the remainder 1 and store to R0

OUT ;Output the remainder 1 as the third value.

BRz ;If exactly zero, then third value is zero 50 -10 (5 times) = 0

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0

OUT ;Display second value on screen

Clear R0

ADD R0,R0, #0

OUT ;Output last digit as zero

;After displaying all three values, continue to display Num8

}

}

Display Eight value

{

Load Num8 to R1

Load ASCII\_OFFSET to R6

Load DC to R5

Check if value is 100, if 0 skip to second value check

{

ADD R2, R2, #-100

ADD R2, R1, R2 ;100-100=0 BRz, 39-100 BRn

BRn

Clear R0 ;If 1st value is zero, print out first value as zero

ADD R0, R0, #0

OUT ;Then jump to 2nd value check

2nd value check ;If value is below 100, enter second value check

BRz ADD R0, R0, #1 ;If BRz, value is 100 so output 100

ADD R0, R1, R0 ;Puts 1 to R0 for output

OUT ;Prints 1 to console

ADD R0, R0, #-1 ;1-1 = 0

OUT ;Then print 0 twice

OUT ;Outputs value "100"

}

Second Value Check

Check if second value is below 10

ADD R2, R1, #-10 ;If value less than 10, then second value is 0 then third value is a single digit value

BRpz DivideLoop ;If more than or equal 10 go to DivideLoop

BRn ;find the 1s value

Clear R0 ;If 2nd value is zero, print out second value as zero

ADD R0, R0, #0

OUT

ADD R0, R6, R1 ;ASCII\_OFFSET the single digit value, store in R0 for display

OUT ;Display Single digit value

DivideLoop ;Divide by ten to find the 10s value

{

ADD R5, R5, #1 ;Add 1 to Division Counter

ADD R4, R1, #0 ;Copy R1 to R4

ADD R4, R4, #-10 ;51/10 is equal to 51- 10(5 times) with remainder 1

BRp DivideLoop ;If still positive, continue divide loop

}

BRn ;If division results in negative, 51-10(6 times) = -11 so

ADD R5, R5, #-1 ;Remove 1 from Divide Counter

ADD R4, R4, #10 ;Place back 10 51-10(5times) = 1

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0 R5= (5 times) so 50s

OUT ;Display second value on screen

ADD R0, R4, R6 ;ASCII\_OFFSET the remainder 1 and store to R0

OUT ;Output the remainder 1 as the third value.

BRz ;If exactly zero, then third value is zero 50 -10 (5 times) = 0

ADD R0, R5, R6 ;ASCII\_OFFSET R5 then store to R0

OUT ;Display second value on screen

Clear R0

ADD R0,R0, #0

OUT ;Output last digit as zero

Halt ;End of program

}

}

}

ERROR Halt ;For error-checking

NumInput1 .STRINGZ "Please input a number value in three digits for sorting (Enter first digit): "

NumInput2 .STRINGZ "Please input a number value in three digits for sorting (Enter second digit): "

NumInput2 .STRINGZ "Please input a number value in three digits for sorting (Enter third digit): "

Num1 .FILL X3100 ;temporary lowest input number

Num2 .FILL X3101

Num3 .FILL X3102

Num4 .FILL X3103

Num5 .FILL X3104

Num6 .FILL X3105

Num7 .FILL X3106

Num8 .FILL X3107 ;temporary highest input number

LC .FILL X0000 ;Loop counter

NC .FILL x0008 ;Counts how many numbers to sort

DC .FILL x0000 ;Division Counter

Inverse\_ASCII\_OFFSET .FILL X-30

ASCII\_OFFSET .FILL X30

I

.END

**Flow chart:**