**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY**

**DEPARTMENT OF TELECOMMUNICATIONS ENGINEERING**

**--------------------------**



**COMPUTING SYSTERMS AND PROGRAMMING  
LC- 3 PROJECT**

**Lecturer: Assoc. Prof. Đặng Thành Tín**

**Students : Đinh Hoàng Luân \_ 2051145**

*HCM City 2021*

**TOPICS**

**PROJECT: LC-3 programming**

This takes you 10% of overall score on my side

------------------------------------------------------  
  
The requirement for the project:  
 1. Use subroutines as much as possible  
 2. Create user interface as clear and beautiful as possible  
 3. Check range for every value input and output appropriately  
 4. The program should be organized so well for structure programming   
 5. The program needs to comment as mamy as possible  
 6. Each student must choose only one topic  
 7. The file .asm of traditional name (StuID\_name…) must be submitted to BKel right before final exam including all supplementing files if you want to make clear more about the project, such as results output, …

------------------------------------------------------

1. Write a program to input *n* (input from keyboard) strings of characters with the length unlimited (it is defined by the program, not by the compiler). Sort them in descending or ascending order depending on the request input.  
   **Attention**:   
    These strings are sorted in ascending/descending by the order of dictionary with deleting redundant characters in each string: blank ‘ ‘, comma ‘,’ if they exist in string.

**1)** **Project's algorithm**

* *I divided my project into* ***6 main*** *parts:*
* **Part 1:** Input n (input from keyboard) strings of characters with the length unlimited and count the number of strings have been input.  
  - I count the number of each strings to control the number of first letter and also use it to sort and remove blank, comma.  
  - I count the characters off each strings and store it in address begin at x5000.  
  - I use “0” in the end of each strings and store it in the next address of the end of each strings, also use like a signal to recognize the next strings
* **Part 2:** Remove redundant blank comma in each strings and print on the screen*.  
  -* While removing redundant blank and comma, I continue to store the character of each new strings in new locations
* **Part 3:** Count the characters in each strings and store it in new locations (x7000) after being removed comma and blank for sorting.
* **Part 4:** Splits the first letter of each string and stores the address of that letter.  
  - Load agian number characters of strings are stored in location begin at x6000 and number characters of each strings are stored in location begin at x7000  
  - Move the address of first letter of each strings to new location
* **Part 5:** Sort them in descending or ascending by the order of dictionary. Print numbers of strings have been sorted following descending the order of dictionary.

**-** I use bubble method to arrange them  
- Load the address of each letter and use function “PUTS” to print out the sequences have been sorted

* **Part 6:** Subroutine  
  - Splits the first letter of each string and stores the address of that letter.  
  - Count the characters in each strings and store it in new locations  
  - Sort in descending   
  - Sort in asencending
* *Show step by step*

***Part 1:*** *Input n (input from keyboard) strings of characters with the length unlimited and count the number of strings have been input.*

.Orig x3000

; Display a stringz "(Input n string)" on the screen

LEA R0, LABEL\_1

PUTS

LABEL\_1 .STRINGZ "(Input n string) \n"

; Label   
 MEMORY\_1 .FILL x5000

ASCII .FILL #48

; Input n (input from keyboard) strings of characters with the length unlimited and count the number of strings have been input  
  
 LD R2, MEMORY\_1

AND R3, R3, #0 ; reset R3

AND R0, R0, #0 ; reset R0

LOOP\_1 GETC ; input characters

ADD R1, R0, #-10

BRz OUTPUT ; input "enter" to have a new string

ADD R1, R1, #-15

ADD R1, R1, #-2

BRz NEXT\_1 ; input "esc" to stop

OUT

STR R0, R2, #0 ; store characters

ADD R2, R2, #1 ; increase address

BR LOOP\_1

OUTPUT AND R0, R0, #0 ; newline

ADD R0, R0, #13 ; newline

OUT

STR R1, R2, #0 ; store "0" at the end of each strings, started at x5000

ADD R2, R2, #1 ; increase address

ADD R3, R3, #1 ; count the number of strings have been input

BR LOOP\_1 ; loop until meet "esc"

NEXT\_1 ST R3, SAVE\_1 ; store the number of strings have been input

AND R0, R0, #0 ; newline

ADD R0, R0, #13 ; newline

OUT

***Part 2:*** *Remove redundant blank comma in each strings and print on the screen.*

; Display a stringz "(The redundant of comma and space have been remove)" on the screen

LEA R0, LABEL\_3

PUTS

LABEL\_3 .STRINGZ "(The redundant of comma and space have been remove) \n"

; Label

Save\_2 .BLKW #1

; Remove redundant blank comma in each strings and print on the screen

AND R3, R3, #0 ; reset R3

AND R4, R4, #0 ; reset R4

LD R5, SAVE\_1 ; load number of sequences

LD R6, MEMORY\_2; load address started at x6000

LD R2, MEMORY\_1; load address started at x5000

LOOP\_3 LDR R1, R2, #0 ; load memory of R2 to R1 load memory of R2 to R3ending

ST R1, SAVE\_2 ; store the chracter

ADD R2, R2, #1 ; increase address

ADD R1, R1, #-15 ; check space

ADD R1, R1, #-15 ; check space

ADD R1, R1, #-2 ; check space

BRz SPACE ; remove redundant space in each strings

AND R4, R4, #0 ; reset R4 to remove abudant space in the next sequences

ADD R1, R1, #-12 ; check comma

BRz COMMA ; remove redundant commma in each strings

AND R3, R3, #0 ; reset R3 to remove abudant space in the next sequences

BR SAVE ; store characters after confirm it not redundant comma or redundant space

SPACE ADD R4, R4, #0 ; accept the legal space

BRp LOOP\_3

ADD R4, R4, #1 ; a signal to deny redundant space

BR SAVE

COMMA ADD R3, R3, #0 ; accept the legal comma

BRp LOOP\_3

ADD R3, R3, #1 ; a signal to deny redundant space

SAVE AND R0, R0, #0

LD R1, SAVE\_2

BRz NEXT\_2 ; if it appear “0” means the sequence has stopped and the next sequence will continue

ADD R0, R1, #0 ; print number of strings have been removed redundant blank and comma

OUT

STR R1, R6, #0 ; store characters or legal space or legal comma

ADD R6, R6, #1 ; increase the address to store a new characters or legal space or legal comma

BR LOOP\_3

NEXT\_2 AND R0, R0 ,#0 ; newline

ADD R0, R0, #13 ; newline

OUT

STR R1, R6, #0 ; store again “0” in the strings

ADD R6, R6, #1 ; increase address

ADD R5, R5, #-1; decrease the number of total strings have been input until it equal to 0

BRz NEXT\_3

BR LOOP\_3

***Part 3:*** *Count the characters in each strings after being removed comma and blank for sorting.*; Count the characters in each string  
  
NEXT\_3 JSR COUNT

***Part 4:*** *Splits the address of the first letter of each string and stores the address of that letter.*

JSR SPLIT

; Sort them in descending or ascending by the order of dictionary

AND R0, R0 ,#0 ; newline

ADD R0, R0, #13 ; newline

OUT ; newline

; LABEL

SAVE\_1 .BLKW #1

MEMORY\_3 .FILL x7000

SAVE\_4 .BLKW #1

MEMORY\_2 .FILL x6000

MEMORY\_1 .FILL x5000

***Part 5:*** *Sort them in descending or ascending by the order of dictiona*ry

; Display a stringz "(Select button 0 for descending and select button 1 for ascending)" on the screen

LEA R0, LABEL\_4

PUTS

LABEL\_4 .STRINGZ "(Select button 0 for descending and select button 1 for ascending) \n"

; Input button 0 for descending and select button 1 for ascending

AND R0, R0, #0

GETC

LD R4, NASCII

ADD R0, R0, R4

BRz DECENDING ; if result is zero move to DECENDING

; Display a stringz "(Ascending the order of dictionary)" on the screen

LEA R0, LABEL\_6

PUTS

LABEL\_6 .STRINGZ "(Ascending the order of dictionary) \n"

; Sort the number of strings in ascending the order of dictionary

JSR MIN

; Print numbers of strings have been sorted following ascending the order of dictionary

AND R0, R0, #0

ADD R0, R0, #13

OUT

LD R3, SAVE\_1 ; load number of sequences

LD R1, MEMORY\_5 ; load address of first letter of each strings

LOOP\_8 LDR R2, R1, #0 ; load first letter of each strings

ADD R0, R2, #0 ; print on screen

PUTS

AND R0, R0, #0 ; newline

ADD R0, R0, #13 ; newline

OUT

AND R0, R0, #0

ADD R1, R1, #1 ; increase address to take the next letter of strings

ADD R3, R3, #-1 ; decrement the number of sequences

BRp LOOP\_8 if result is positive mean the number of strings is stil loop

BR STOP ; stop the program

; Display a stringz "(Descending the order of dictionary)" on the screen

DECENDING LEA R0, LABEL\_5

PUTS

LABEL\_5 .STRINGZ "(Descending the order of dictionary) \n"

; Sort the number of strings in descending the order of dictionary

JSR MAX

; Print numbers of strings have been sorted following desending the order of dictionary

AND R0, R0, #0 ; newline

ADD R0, R0, #13 ; newline

OUT

LD R3, SAVE\_1 ; load number of sequences

LD R1, MEMORY\_5 ; load address of first letter of each strings

LOOP\_9 LDR R2, R1, #0 ; load first letter of each strings

ADD R0, R2, #0 ; print on screen

PUTS

AND R0, R0, #0 ; newline

ADD R0, R0, #13 ; newline

OUT

AND R0, R0, #0

ADD R1, R1, #1 ; increase address to take the next letter of strings

ADD R3, R3, #-1 ; decrement the number of sequences

BRp LOOP\_9 ; if result is positive mean the number of strings is stil loop

STOP HALT

***Part 6:*** *Subroutine*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

COUNT LD R1, MEMORY\_2 ; load character of each strings have been removed comma and space

LD R3, SAVE\_1 ; load the number of strings

LD R5, MEMORY\_3 ; prepare the address to store number of characters of each strings after being removed comma and space

LOOP\_5 AND R4, R4, #0

LOOP\_4 LDR R2, R1, #0 ; load characters from Memory\_3

ADD R2, R2, #0

BRz NEXT\_4 ; if R2 = “0” it means that is the end of the strings and move to the next strings

ADD R4, R4, #1 ; count the characters

ADD R1, R1, #1 ; increase address to load next characters

BR LOOP\_4

NEXT\_4 STR R4, R5, #0 ; store number characters of each strings in location x7000

ADD R5, R5, #1 ; increase address to store characters

ADD R1, R1, #1 ; increase address to load next characters

ADD R3, R3, #-1 ; decrease total of strings have been input

BRp LOOP\_5 ; if R3 still positive it continue

;###############################################################

; Splits the first letter of each string and stores the address of that letter  
SPLIT LD R1, MEMORY\_2 ; number characters of strings are stored in location x6000

LD R2, MEMORY\_3 ; number characters of each strings are stored in location x7000

LD R3, SAVE\_1 ; load number of strings

LD R6, MEMORY\_5 ; prepare the address to store address of first letter of each strings

LOOP\_7 ADD R3, R3, #-1 ; decrease number of strings

BRn NEXT\_6 ; until it negative

LDR R4, R1, #0 ; load characters

ST R1, SAVE\_4 ; store address of first letter of each strings

LDR R5, R2, #0 ; load number of characters of each strings

LOOP\_6 ADD R1, R1, #1 ; increase address

ADD R5, R5, #-1 ; decrease number of characters

BRp LOOP\_6 ; it positive then continue to count

LD R5, SAVE\_4 ; if it negative it means that the next string has appeared, then load address of first letter of each strings again

ADD R1, R1, #1 ; increase address

STR R5, R6, #0 ; store the address of first letter of each string in location x9000

ADD R6, R6, #1 ; increase the location to store new address of first letter

ADD R2, R2, #1 ; continue increase the address of number characters of each strings

BR LOOP\_7

NEXT\_6 RET

;###############################################################MIN LD R4, SAVE\_1 ; load number of strings

OUTER ADD R4, R4, #-1 ; loop n - 1 times

BRnz NEXT\_5 ; looping complete, exit

ADD R6, R4, #0 ; initialize inner loop counter to outer

LD R1, MEMORY\_5 ; set file pointer to begin of file

INNER LDR R2, R1, #0 ; load address contain address of first letter of strings

LDR R0, R2, #0 ; load the first letter

LDR R3, R1, #1 ; load address contain address of first letter of the next strings

LDR R5, R3, #0 ; load the first letter of the next strings

ST R5, SAVE\_5

NOT R5, R5 ; compare two first letter

ADD R5, R5, #1

ADD R5, R0, R5

BRnz SWAP ; if the result positive change the address of two first letter

LD R5, SAVE\_5

STR R3, R1, #0

STR R2, R1, #1

SWAP LD R5, SAVE\_5

ADD R1, R1, #1 ; increment file pointer

ADD R6, R6, #-1 ; decrement inner loop counter

BRp INNER ; end of inner loop

BR OUTER ; end of outer loop

NEXT\_5 RET

;###############################################################

MAX LD R4, SAVE\_1 ; load number of strings

OUTER\_1 ADD R4, R4, #-1 ; loop n - 1 times

BRnz NEXT\_7 ; looping complete, exit

ADD R6, R4, #0 ; initialize inner loop counter to outer

LD R1, MEMORY\_5 ; set file pointer to begin of file

INNER\_1 LDR R2, R1, #0 ; load address contain address of first letter of strings

LDR R0, R2, #0 ; load the first letter

LDR R3, R1, #1 ; load address contain address of first letter of the next strings

LDR R5, R3, #0 ; load the first letter of the next strings

ST R5, SAVE\_5

NOT R5, R5 ; compare two first letter

ADD R5, R5, #1

ADD R5, R0, R5

BRp SWAP\_1 ; if the result negative change the address of two first letter

LD R5, SAVE\_5

STR R3, R1, #0

STR R2, R1, #1

SWAP\_1 LD R5, SAVE\_5

ADD R1, R1, #1 ; increment file pointer

ADD R6, R6, #-1 ; decrement inner loop counter

BRp INNER\_1 ; end of inner loop

BR OUTER\_1 ; end of outer loop

NEXT\_7 RET

; LABEL

MEMORY\_5 .FILL x9000

NASCII .FILL #-48

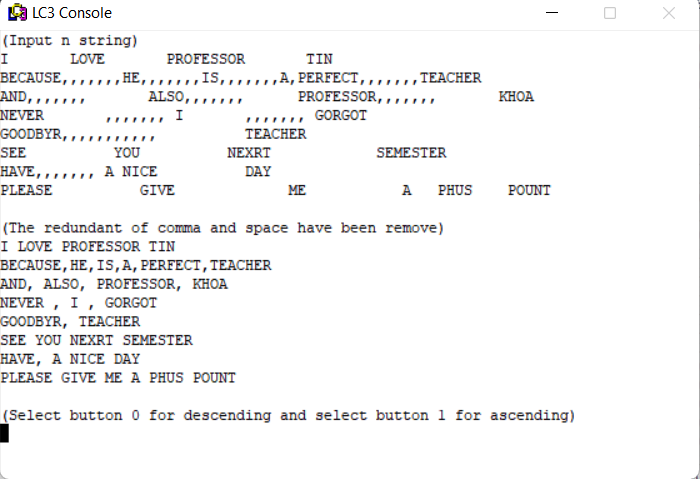
SAVE\_3 .BLKW #1

SAVE\_5 .BLKW #1

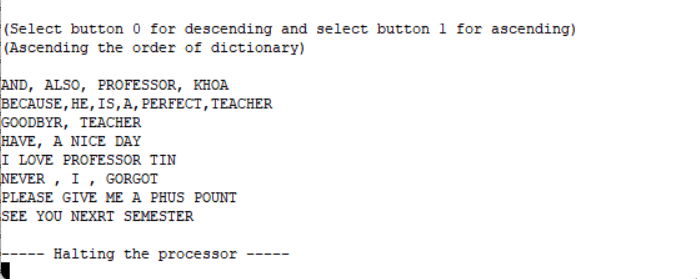
.End

**2) Demontration**

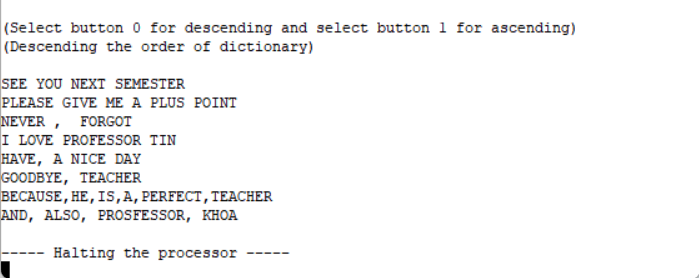
* *Input and remove redundant space and comma*



* *Output result following ascending*



* *Output result following descending*

**

**3) Operate code**

* ***Step 1***: Input the initial string you want then press “enter” to input the next string from the keyboard.
* ***Step 2:*** In order to halt the string that you are input ,you press “enter” after the last string, then press “esc” immediately.
* ***Step 3:*** You press “0” for descending or press “1” for ascending.