MINIPROJECT REPORT

# CYCLONE WORD PROGRAM

1. **Problem statement**

* Cyclone words are English words that have a sequence of characters in alphabetical order when following a cyclic pattern
* Take an input string from user then check if it is a cyclone word or not

1. **Used labels & registers in main**
   1. **Labels**

string: input string of maximum length 100

solved\_string: input string after being converted to cyclone form

length: store length of input string

i: counter index for string, default = 0

i1: counter index for solved\_string, default = 0

loop\_limit: limiting loop for range, default = 0

isOdd: return (length(string) %2)

immediate: index for finding character purpose, immediate = length(string) – 1 - i

validMessage: “ is a cyclone phrase” is displayed when input string is cyclone

invalidMessage: “ is not a cyclone phrase” is displayed when input string is not cyclone

* 1. **Registers**

$v0: contains length value, syscall purpose

$t0, $t1: temporary storage register

$a0: contain address of the character at solved\_string[i1]

$a1: contain address of the character at string[i]

1. **Function’s Algorithm**
   1. **main**

* initialize solved\_string = null, isOkay = 1 (True)
* get string from user
* get the length of the string by using “getLengthLabel” function
* the first for-loop to scan all characters in input string and then store each of them in solved\_string in cyclone form, (from I = 0 to length(string)/2), inside for-loop:

+ call “getAddressAtIndex” function to get character string[i] and string[length(string)-i-1]

+ set solved\_string[i1] = string[i], solved\_string[i1+1] = string[length(string)-i-1] by using “copyChar” function, then increase index i1 by 1 for the next loop

* if length(string) is odd, then append string[i] (the middle character of input string) to solved\_string
* the second for-loop to scan all characters in solved\_string from beginning to the last character (from i = 0 to length(string)-2), inside for-loop:

+ call “getAddressAtIndex” function to get character solved\_string[i] and solved\_string[i+1]

+ if solved\_string[i] < solved\_string[i+1] go to the next loop

+ else set isOkay = 0 (False) then break the loop

* print the string and show if it is a surpassing word or not by using value inside “isOkay”
  1. **getLengthLabel(%label)**
  2. **getAddressAtIndex(%str, %index)**
  3. **getCharDistance(%register\_1, %register\_2)**
  4. **copyChar(%destination, %source)**

*Please read MINIPROJECT SUBPROGRAM CATALOG for details of these 4 subprograms.*

1. **Pseudocode**

string word; (input)

string solString = null;

int flag = 1;

#convert input string to cyclone form

for (int i = 0 ; i < n/2 ; i++){

solString.append(word[i]);

solString.append(word[length(word)-1-i));

}

if (length(word) % 2 != 0){

solString.append(word[i])

}

#test cyclone phrase

for (int i = 0 ; i < n-1 ; i++){

char x = str[i];

char y = str[i+1];

if (y >= x)

continue;

else{

flag = 0;

break;

}

}

if (flag == 0)

printf("is\_cyclone\_phrase(\"%s\")# => False", str);

else printf("is\_cyclone\_phrase(\"%s\")# => True", str);