Republic of the Philippines
CAMARINES SUR POLYTECHNIC COLLEGES
Nabua, Camarines Sur

COLLEGE *of* COMPUTER STUDIES

CS 318 – Architecture and Organization

LEARNING TASK (STRINGS)

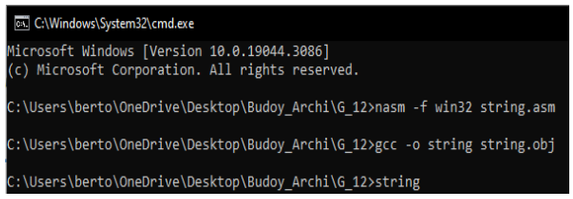
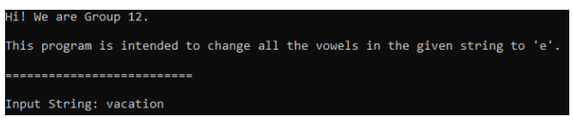
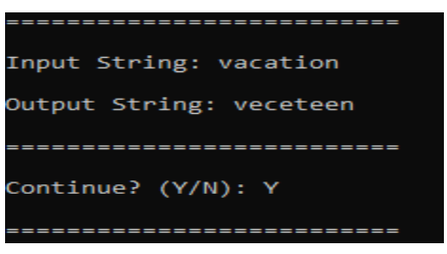
GROUP NO: 12 SECTION: BSCS-3A

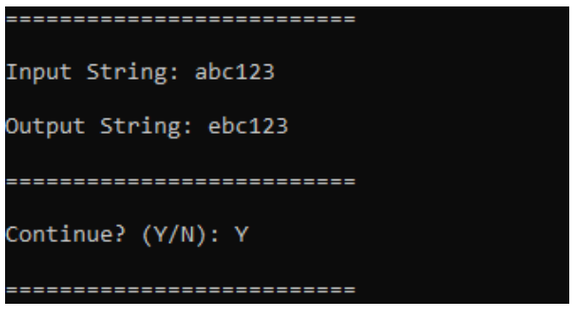
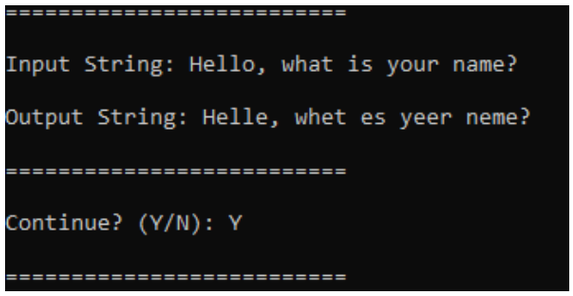
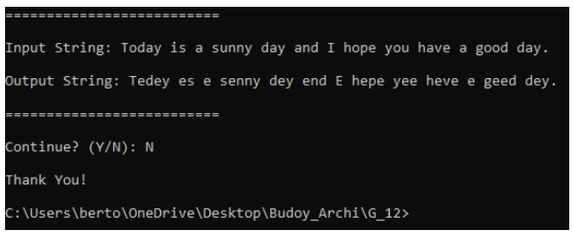
GROUP MEMBERS: Roberto Bayos Jr.
John Peter Alcoy
Marc Christian Tumaneng

SAMPLE RUN

Step-by-step sample run of your assembly program with explanation.

YouTube link: *https://youtu.be/nUDbTite_CE*

Step 1 As usual, Assemble the **"string.asm"** assembly language source code using the NASM assembler with the target output format set to win32. Next the GCC compiler… compile and link an object file named **"string.obj"** to get an executable program named ***string***.	*Image 1* 
Step 2 Now, after executing the string assembly program, it will show the introductory message ***"Hi! We are Group 12."*** The display prompt message ***"This program is intended to change all the vowels in the given string to 'e'."*** As well as the ***"Input String:"*** that will serve as the message where the user will write to in order to get the first initial output string.	*Image 2* 
Step 3 So, as you can see after typing the word ***"vacation"*** the output string changed all the vowels into "e". and after that, the user asked if he/she wants to continue by choosing whether **(Y/N).**	*Image 3* 

Step 4 Now that we chose to continue the program, it asked again to type a string, this time we put ***"abc123"*** and as usual the vowels which is the ***"a"*** changed into ***"e"*** while the rest letters, and numbers will remain as it is. Then asked again by the program if we want to continue.	*Image 4* 
Step 5 In this case we have a total of 5 words and also the punctuation marks, and so again the vowels will be changed to ***"e"*** while the rest will remain unchanged including the punctuation marks and the capital letter ***"H"***.	*Image 5* 
Step 6 We're now in the last step, in this case the letters that are not vowels are still unchangeable, and the letters that considered a vowel will be changed to ***"e"*** including the capital letter ***"A".*** Now, to exit the program just type ***"N" / "n"*** to close/exit the program.	*Image 6* 

ASSEMBLERS INSTRUCTIONS

String Instructions
List the String Instructions you have use and define its purpose. ***lodsb:*** This instruction loads the byte at the address stated in the esi register into the al register, increments esi to point to the next byte, and uses the result to set the Zero Flag (ZF). It loads characters from the input string. ***stosb:*** This instruction stores the byte at the address specified in the edi register in the al register, increments edi so that it points to the next byte, and updates flags. It stores characters in the output string. ***lea:*** The effective address of the source or location operand is loaded into the specified register with this instruction. This code says: ***lea esi, [input_str]:*** The address of the input string is loaded into the esi register. ***lea edi, [output_str]:*** The address of the output string is loaded into the edi register. ***cmp:*** This command compares two operands. ***jmp (Jump):*** This instruction unconditionally transfers control to the specified target address. ***je (Jump if Equal):*** If the Zero Flag (ZF) has been set, showing that the two compared operands are equal, this command jumps to the specified target address. ***jl (Jump if Less) and jg (Jump if Greater):*** If the outcome of a signed comparison is smaller or larger than zero, these instructions will jump to the specified target address. ***mov:*** This instruction transfers data from the source to the location operand. This code says: The letter ***'e':*** is used to replace a vowel.

The command mov dword [edi], 0: places a null terminator at the end of the output string.

add: This instruction combines the source and destination operands and stores the result in the operand that is to be combined.

push: The specified value is pushed onto the stack by this instruction.

pop: The value to the top of a stack is popped into the specified destination using this instruction.

PROGRAM CODE

```
Insert your assembly (NASM) code here, be sure to add comments to
describe each line/set of codes...

;Group No. 12

;Design an Assembly program to change all the vowels in the given string to 'e'.
;Include section labels on your code and comments explaining each line and at least
one string instruction in your assembly code.
;The program loops until the user chooses N to Exit.

section .data
   prompt_welcome db "Hi! We are the Group 12", 10, "This program is intended to
change all the vowels in the given string to 'e'.", 10, 0
   prompt_input   db "Input String: ", 0
   instr db '%s', 0
   prompt_output  db "Output String: ", 0
   outstr db '%s', 0
   prompt_continue db "Continue? (Y/N): ", 0
   choice db '%s', 0
   vowels db 'aeiouAEIOU', 0  ; String containing vowels
   exit_msg db "Thank You!", 10, 0
   error_msg db "===== Error! Invalid input. Please enter 'Y' or 'N' ======", 10, 0
   line db "==============================", 10, 0

newline db  10, 0


section .bss

   input_str resb 256
   output_str resb 256
   user_choice resb 2
```

```
section .text
    global _main
    extern _printf
    extern _scanf
    extern _gets
    extern _exit

error_function:
    push error_msg
    call _printf
    add esp, 4
    ret


_main:
    ; Display welcome message
    push prompt_welcome
    call _printf
    add esp, 4

main_loop:
    push line
    call _printf
    add esp, 4

    ; Get user input
    push prompt_input
    call _printf
    add esp, 4

    push input_str
    push  instr
    call _scanf
    add esp, 8
```

```
    ; Process the input string
    lea esi, [input_str]  ; Load address of input string into esi
    lea edi, [output_str] ; Load address of output string into edi


process_string_loop:
lodsb  ; Load character from esi into AL, and increment esi
cmp al, 0  ; Check if end of string
je end_process_string
cmp al, 'a'
je replace_with_e
cmp al, 'e'
je replace_with_e
cmp al, 'i'
je replace_with_e
cmp al, 'o'
je replace_with_e
cmp al, 'u'
je replace_with_e
cmp al, 'A'
je replace_with_e
cmp al, 'E'
je replace_with_e
cmp al, 'I'
je replace_with_e
cmp al, 'O'
je replace_with_e
cmp al, 'U'
je replace_with_e
jmp copy_character

replace_with_e:
    cmp al, 'a'
    jl not_lower_vowel
```

```
    cmp al, 'z'
    jg not_lower_vowel
    ; It's a lowercase vowel, replace with 'e'
    mov al, 'e'
    jmp after_replace

not_lower_vowel:
    cmp al, 'A'
    jl not_upper_vowel
    cmp al, 'Z'
    jg not_upper_vowel
    ; It's an uppercase vowel, replace with 'E'
    mov al, 'E'
    jmp after_replace

not_upper_vowel:
    ; Not a vowel, do not replace
    jmp copy_character

after_replace:
    stosb
    jmp process_string_loop

copy_character:
    stosb
    jmp process_string_loop

end_process_string:
    mov byte [edi], 0

    ; Display the output string
    push prompt_output
    call _printf
    add esp, 4
```

```
    push output_str
    push  outstr
    call _printf
    add esp, 8

    push newline
    call _printf
    add esp, 4

continue_prompt:
    push line
    call _printf
    add esp, 4

    ; Ask if the user wants to continue
    push prompt_continue
    call _printf
    add esp, 4

    push user_choice
    push choice
    call _scanf
    add esp, 8

    ; Consume remaining characters in the input buffer
    push input_str
    call _gets
    add esp, 4

    cmp byte [user_choice], 'N'
    je exit
    cmp byte [user_choice], 'n'
    je exit

    cmp byte [user_choice], 'Y'
```

```
    je next_iteration
    cmp byte [user_choice], 'y'
    je next_iteration

    ; Display an error message and go back to continue_prompt
    call error_function
    jmp continue_prompt

next_iteration:
    ; Add a newline before restarting the loop
    jmp main_loop

exit:
    ; Display thank you message and exit
    push exit_msg
    call _printf
    add esp, 4

    call _exit
```

GROUP ASSIGNMENT

NAME OF MEMBER	TASK ACCOMPLISHED
Roberto Bayos Jr.	coding, documentation, video recording
Marc Christian Tumaneng	coding, documentation, video recording
John Peter Alcoy	coding, documentation, video recording