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

COLLEGE *of* COMPUTER STUDIES

CS 318 – Architecture and Organization

LEARNING TASK (SAL PART 2 – MODULAR PROGRAMMING)

GROUP NO: 12 SECTION: BSCS-3A

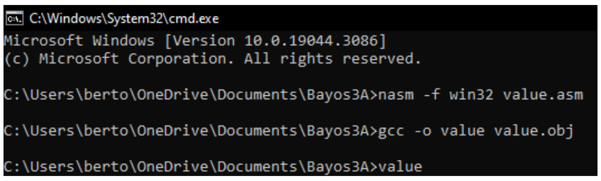
GROUP MEMBERS: Marc Christian Tumaneng
John Peter Alcoy
Roberto Bayos Jr.

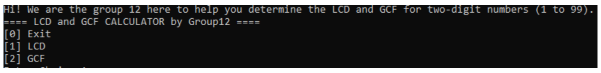
SAMPLE RUN

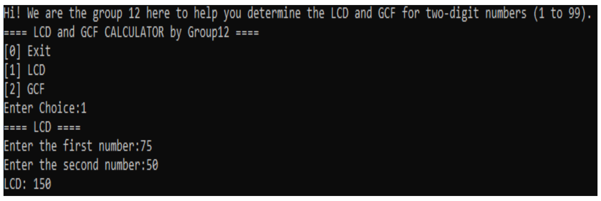
Step-by-step sample run of your assembly programs with explanation.

YouTube link/s: *https://youtu.be/1jhKw492-po*

Pass-by-Value

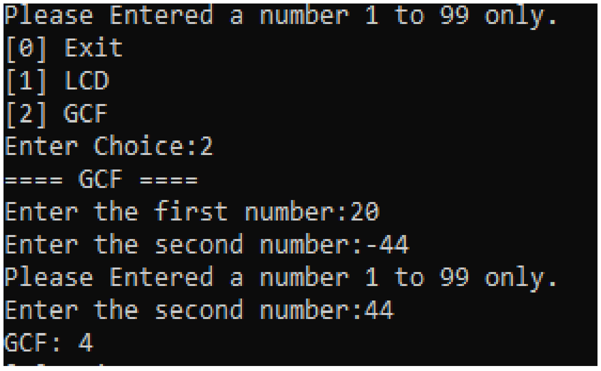
Step 1
As usual, Assemble the **"value.asm"** assembly language source code using the NASM assembler with the target output format set to win32. Next the GCC compiler… compile and link an object file named "value.obj" to get an executable program named ***value***.
	Image 1
	
	C:\Windows\System32\cmd.exe
	`Microsoft Windows [Version 10.0.19044.3086]`
	`(c) Microsoft Corporation. All rights reserved.`
	`C:\Users\berto\OneDrive\Documents\Bayos3A>nasm -f win32 value.asm`
	`C:\Users\berto\OneDrive\Documents\Bayos3A>gcc -o value value.obj`
	`C:\Users\berto\OneDrive\Documents\Bayos3A>value`

Step 2
Now, after executing the LCD and GCF Calculator in pass-by-value program, it will show the introductory message "Hi! We are the group 12 here to help you determine the LCD and GCF for two-digit numbers (1 to 99).'' "==== LCD and GCF CALCULATOR by Group12 ====". Next it shows the menu which the users be able choose what he/she wants to use for calculation. It can be LCD, or GCF.
	Image 2
	
	`Hi! We are the group 12 here to help you determine the LCD and GCF for two-digit numbers (1 to 99).`
	`==== LCD and GCF CALCULATOR by Group12 ====`
	`[0] Exit`
	`[1] LCD`
	`[2] GCF`

Step 3
As you can see, we first get the result for the LCD, finding computing between 75 and 50 which result of 150.
	Image 3
	
	`Hi! We are the group 12 here to help you determine the LCD and GCF for two-digit numbers (1 to 99).`
	`==== LCD and GCF CALCULATOR by Group12 ====`
	`[0] Exit`
	`[1] LCD`
	`[2] GCF`
	`Enter Choice:1`
	`==== LCD ====`
	`Enter the first number:75`
	`Enter the second number:50`
	`LCD: 150`

Step 4	*Image 4*

Now, moving to getting the result for GCF, and as you noticed in the program only 1-99 numbers only the users can able to input, not 0 nor negative numbers and above 99… and in this case the user typed -44 which not accepted by the program, that's why it asked for another number which is only a valid one, and it's 44 which is accepted. Now we'll get the result for the GCF "4".

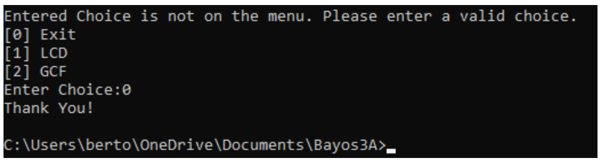


```
Please Entered a number 1 to 99 only.
[0] Exit
[1] LCD
[2] GCF
Enter Choice:2
==== GCF ====
Enter the first number:20
Enter the second number:-44
Please Entered a number 1 to 99 only.
Enter the second number:44
GCF: 4
```

Step 5

Last one, the exit program when type 0 the program will end with the "Thank you" message.

Image 5



```
Entered Choice is not on the menu. Please enter a valid choice.
[0] Exit
[1] LCD
[2] GCF
Enter Choice:0
Thank You!

C:\Users\berto\OneDrive\Documents\Bayos3A>
```

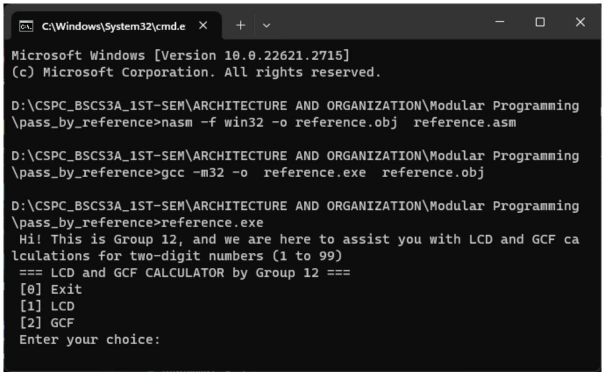
Pass-by-Reference

step 1

use the following commands to run the program:

- *nasm -f win32 -o reference.obj reference.asm*
- *gcc -m32 -o reference.exe reference.obj*
- *reference.exe*

image 1

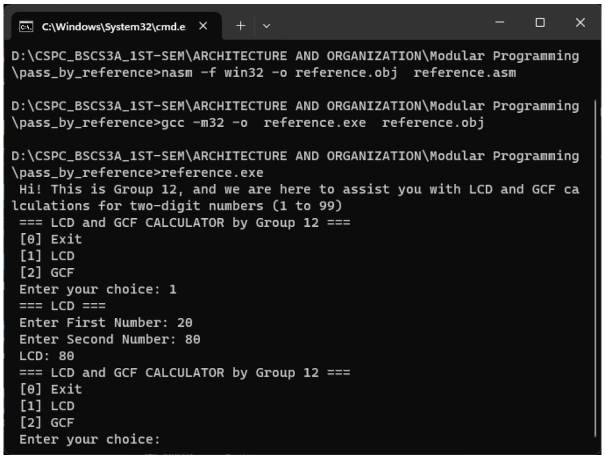


```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

D:\CSPC_BSCS3A_1ST-SEM\ARCHITECTURE AND ORGANIZATION\Modular Programming
\pass_by_reference>nasm -f win32 -o reference.obj  reference.asm

D:\CSPC_BSCS3A_1ST-SEM\ARCHITECTURE AND ORGANIZATION\Modular Programming
\pass_by_reference>gcc -m32 -o  reference.exe  reference.obj

D:\CSPC_BSCS3A_1ST-SEM\ARCHITECTURE AND ORGANIZATION\Modular Programming
\pass_by_reference>reference.exe
 Hi! This is Group 12, and we are here to assist you with LCD and GCF ca
lculations for two-digit numbers (1 to 99)
 === LCD and GCF CALCULATOR by Group 12 ===
 [0] Exit
 [1] LCD
 [2] GCF
 Enter your choice:
```

step 2

- ***select 1 for lcd calculation***
- *the program will ask for the first and second number to be entered, after that it will display the result*
- *the program will continue to loop until user select 0*

image 2

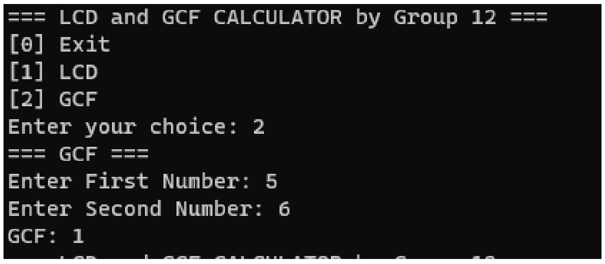


```
D:\CSPC_BSCS3A_1ST-SEM\ARCHITECTURE AND ORGANIZATION\Modular Programming
\pass_by_reference>nasm -f win32 -o reference.obj  reference.asm

D:\CSPC_BSCS3A_1ST-SEM\ARCHITECTURE AND ORGANIZATION\Modular Programming
\pass_by_reference>gcc -m32 -o  reference.exe  reference.obj

D:\CSPC_BSCS3A_1ST-SEM\ARCHITECTURE AND ORGANIZATION\Modular Programming
\pass_by_reference>reference.exe
 Hi! This is Group 12, and we are here to assist you with LCD and GCF ca
lculations for two-digit numbers (1 to 99)
 === LCD and GCF CALCULATOR by Group 12 ===
 [0] Exit
 [1] LCD
 [2] GCF
 Enter your choice: 1
 === LCD ===
 Enter First Number: 20
 Enter Second Number: 80
 LCD: 80
 === LCD and GCF CALCULATOR by Group 12 ===
 [0] Exit
 [1] LCD
 [2] GCF
 Enter your choice:
```

step 3

- ***select 2 for gcf calculation***
- *the program will ask for the first and second number to be entered, after that it will display the result*
- *the program will continue to loop until user select 0*

image 3

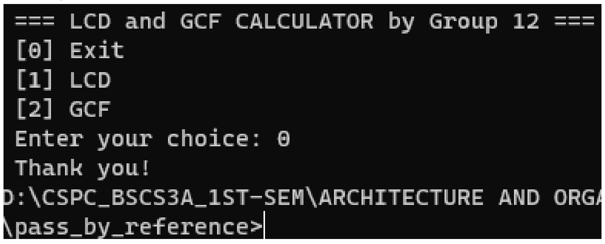


```
=== LCD and GCF CALCULATOR by Group 12 ===
[0] Exit
[1] LCD
[2] GCF
Enter your choice: 2
=== GCF ===
Enter First Number: 5
Enter Second Number: 6
GCF: 1
```

step 4

- ***select 0 to exit the program***

image 4



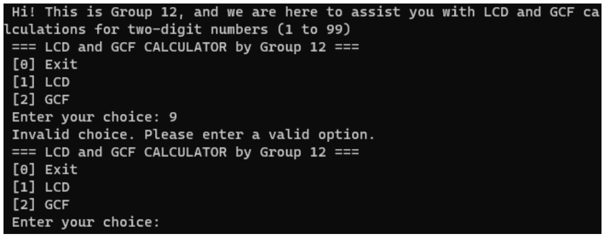
```
 === LCD and GCF CALCULATOR by Group 12 ===
 [0] Exit
 [1] LCD
 [2] GCF
 Enter your choice: 0
 Thank you!
D:\CSPC_BSCS3A_1ST-SEM\ARCHITECTURE AND ORGA
\pass_by_reference>
```

step 5

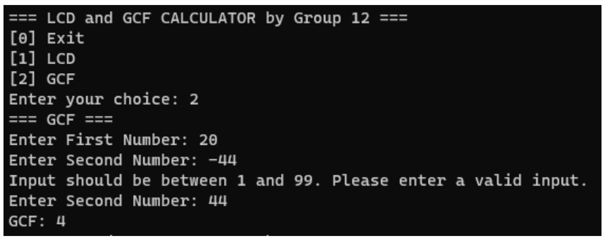
error handling/ input validation

- *if the user enters a number that is not in the menu, it will keep on asking until the user enters the correct input*
- *if the user enter a number that do not belong to numbers 1 to 99, it will keep on asking until the user enters the correct input*

image 5



```
 Hi! This is Group 12, and we are here to assist you with LCD and GCF ca
lculations for two-digit numbers (1 to 99)
 === LCD and GCF CALCULATOR by Group 12 ===
 [0] Exit
 [1] LCD
 [2] GCF
 Enter your choice: 9
 Invalid choice. Please enter a valid option.
 === LCD and GCF CALCULATOR by Group 12 ===
 [0] Exit
 [1] LCD
 [2] GCF
 Enter your choice:
```

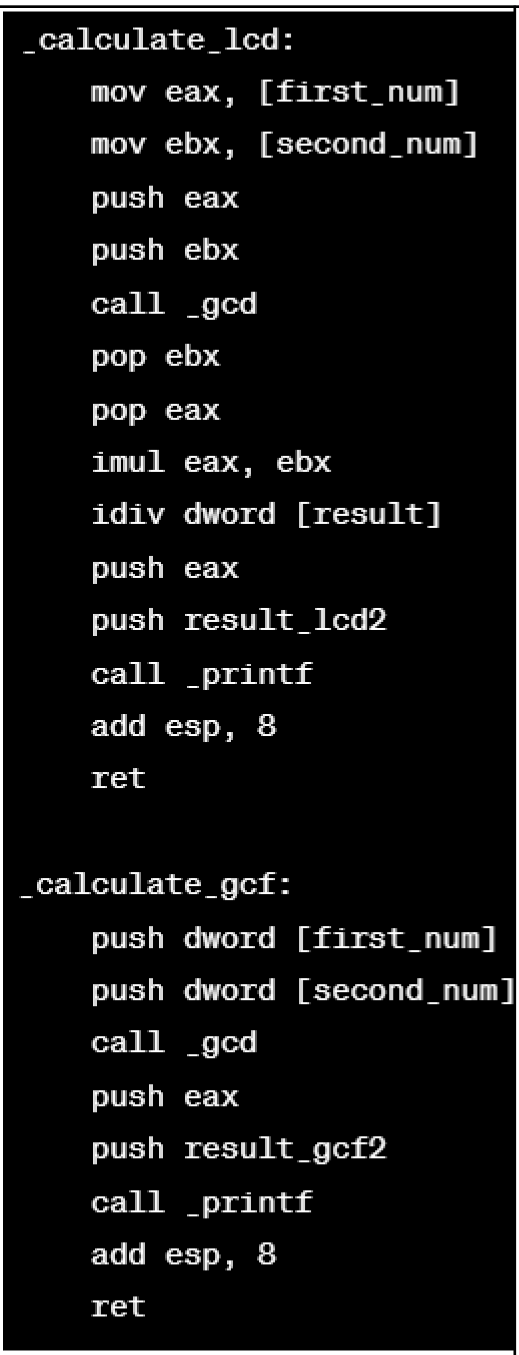


```
=== LCD and GCF CALCULATOR by Group 12 ===
[0] Exit
[1] LCD
[2] GCF
Enter your choice: 2
=== GCF ===
Enter First Number: 20
Enter Second Number: -44
Input should be between 1 and 99. Please enter a valid input.
Enter Second Number: 44
GCF: 4
```

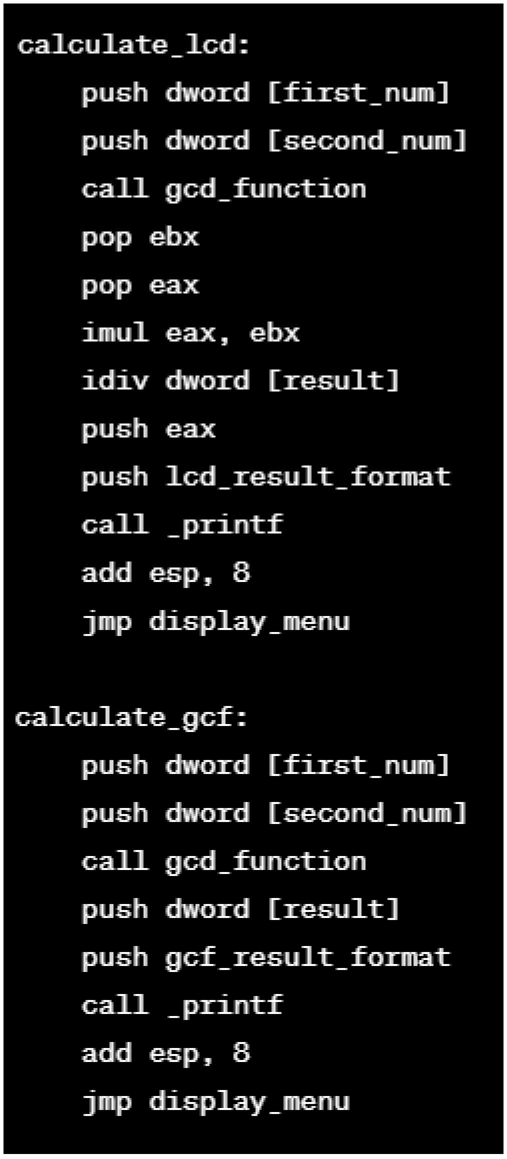
PASS-BY-VALUE VS PASS-BY-REFERENCE

Compare your two assembly programs. Illustrate and explain your given comparison.

Pass-by-Value	***Pass-by-Reference***
Approach to Passing Parameters - *Values are explicitly loaded into registers before the function call.* - *The GCD function is called independently for both LCD and GCF calculations.* - *The results are then used in the calculation.* - *More manual control over data movement.* - *The function cannot accidentally change the original variable.* - *Changes to the parameter within the function have no effect on the variable outside the function.*	*Approach to Passing Parameters* - *Memory addresses (references) are directly pushed onto the stack before the function call.* - *The GCD function is common for both LCD and GCF calculations.* - *Results are stored at memory addresses pointed by references.* - *Cleaner separation of concerns between GCD calculation and result handling.* - *Instead of giving the actual value, a reference (memory address) to the variable is supplied to the function in pass by reference.*



```
_calculate_lcd:
    mov eax, [first_num]
    mov ebx, [second_num]
    push eax
    push ebx
    call _gcd
    pop ebx
    pop eax
    imul eax, ebx
    idiv dword [result]
    push eax
    push result_lcd2
    call _printf
    add esp, 8
    ret

_calculate_gcf:
    push dword [first_num]
    push dword [second_num]
    call _gcd
    push eax
    push result_gcf2
    call _printf
    add esp, 8
    ret
```



```
calculate_lcd:
    push dword [first_num]
    push dword [second_num]
    call gcd_function
    pop ebx
    pop eax
    imul eax, ebx
    idiv dword [result]
    push eax
    push lcd_result_format
    call _printf
    add esp, 8
    jmp display_menu

calculate_gcf:
    push dword [first_num]
    push dword [second_num]
    call gcd_function
    push dword [result]
    push gcf_result_format
    call _printf
    add esp, 8
    jmp display_menu
```

PROGRAM CODE

Pass-by-Value

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

```
section .data
   ; Welcome message and menu prompts
   prompt_welcome db 'Hi! We are the group 12 here to help you determine the LCD and GCF for two-digit numbers (1 to 99).',10,0
   prompt_display db '==== LCD and GCF CALCULATOR by Group12 ====',10, 0
   prompt_exit db  '[0] Exit',10, 0
   prompt_lcd db   '[1] LCD',10, 0
   prompt_gcf db   '[2] GCF',10, 0
   selectChoice db 'Enter Choice:', 0
   inputformat_selectChoice db '%d', 0

   ; Prompts for entering numbers
   prompt_first_num db 'Enter the first number:', 0
   prompt_second_num db 'Enter the second number:', 0
   inputformat_numbers db '%d', 0

   ; Result messages
   result_lcd2 db 'LCD: %d',10, 0
   result_gcf2 db 'GCF: %d',10, 0

   ; Error messages
   prompt_invalid_choice db 'Entered Choice is not on the menu. Please enter a valid choice.',10, 0
   choice_num db 'Please Entered a number 1 to 99 only.',10, 0

   ; Exit message
   exit db 'Thank You!', 10, 0

   ; Section headers for LCD and GCF
   prompt_lcd3 db '==== LCD ====', 10, 0
```

```
    prompt_gcf3 db '==== GCF ====', 10, 0

section .bss
   ; Variables to store user input and results
   menu_choice resb 100
   first_num resd 1
   second_num resd 1
   operatorchoice resd 1
   result resd 1  ; To store the result

section .text
global _calculate_gcf  ; Function to calculate GCF
global _calculate_lcd  ; Function to calculate LCD (pass-by-value)
global _main
extern _printf
extern _scanf
extern _exit

; GCD calculation function
_gcd:
    push ebp
    mov ebp, esp
    mov eax, [first_num]  ; First number
    mov ebx, [second_num]  ; Second number
    gcd_loop:
        cmp ebx, 0
        je gcd_done
        xor edx, edx
        div ebx
        mov eax, ebx
        mov ebx, edx
        jmp gcd_loop
    gcd_done:
    mov [result], eax  ; Store the result
    pop ebp
    ret

; LCD calculation function
_calculate_lcd:
    mov eax, [first_num]
    mov ebx, [second_num]
    push eax  ; Push the first number
    push ebx  ; Push the second number
    call _gcd

    pop ebx
    pop eax
    imul eax, ebx  ; Multiply the numbers
    idiv dword [result]  ; Divide by the GCD result

    ; Display the result
    push eax
    push result_lcd2
    call _printf
    add esp, 8
    ret

; GCF calculation function
_calculate_gcf:
    .gcf_loop:
        cmp ebx, 0
        je .done
        xor edx, edx
        div ebx
        mov eax, ebx
        mov ebx, edx
        jmp .gcf_loop
    .done:
        ret
```

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

    ; Display program title
    push prompt_display
    call _printf
    add esp, 4

.menu_loop:
    ; Display menu options
    push prompt_exit
    call _printf
    add esp, 4

    push prompt_lcd
    call _printf
    add esp, 4

    push prompt_gcf
    call _printf
    add esp, 4

    ; Display menu prompt and get user choice
    push selectChoice
    call _printf
    add esp, 4

    ; Read menu choice
    push menu_choice
    push inputformat_selectChoice
    call _scanf
    add esp, 8

   ; Check the menu choice
    mov eax, [menu_choice]
    cmp eax, 1 ; LCD
    je  .lcd_calculation
    cmp eax, 2 ; GCF
    je  .gcf_calculation
    cmp eax, 0 ; Exit
    je .exit

    ; Invalid choice, loop again
    push prompt_invalid_choice
    call _printf
    add esp, 4
    jmp .menu_loop

.exit:
    ; Display exit message and exit the program
    push  exit
    call _printf
    call _exit

.lcd_calculation:
    ; Display LCD section title
    push prompt_lcd3
    call _printf
    add esp, 4

    ; Read the first number
    push prompt_first_num
    call _printf
    add esp, 4

    ; Read and validate the first number
```

```
.read_first:
push first_num
push inputformat_numbers
call _scanf
add esp, 8

; Check if the first number is in the valid range (1 to 99)
mov eax, [first_num]
cmp eax, 1
jl .invalid_number
cmp eax, 99
jg .invalid_number

; Read the second number
push prompt_second_num
call _printf
add esp, 4

; Read and validate the second number
.read_second:
push second_num
push inputformat_numbers
call _scanf
add esp, 8

; Check if the second number is in the valid range (1 to 99)
mov ebx, [second_num]
cmp ebx, 1
jl .invalid_number
cmp ebx, 99
jg .invalid_number

; Calculate LCD using the function
call _calculate_lcd
```

```
    .invalid_number:
    ; Display error message for invalid number and loop to the menu
    push choice_num
    call _printf
    add esp, 4
    jmp .menu_loop

.gcf_calculation:
    ; Display GCF section title
    push prompt_gcf3
    call _printf
    add esp, 4

    ; Read the first number
    push prompt_first_num
    call _printf
    add esp, 4

    ; Read and validate the first number
    .read_first_gcf:
    push first_num
    push inputformat_numbers
    call _scanf
    add esp, 8

    ; Check if the first number is in the valid range (1 to 99)
    mov eax, [first_num]
    cmp eax, 1
    jl .invalid_number_gcf
    cmp eax, 99
    jg .invalid_number_gcf

    ; Read the second number
    push prompt_second_num
    call _printf
```

```
add esp, 4

; Read and validate the second number
.read_second_gcf:
push second_num
push inputformat_numbers
call _scanf
add esp, 8

; Check if the second number is in the valid range (1 to 99)
mov ebx, [second_num]
cmp ebx, 1
jl .invalid_number_gcf
cmp ebx, 99
jg .invalid_number_gcf

; Check if both numbers are positive
cmp eax, 0
jle .invalid_number_gcf
cmp ebx, 0
jle .invalid_number_gcf

; Perform GCF calculation with the valid numbers
push dword [first_num]
push dword [second_num]
mov eax, [esp+4]
mov ebx, [esp+8]

; Call GCF function
call _calculate_gcf

; Display the result
push eax
push result_gcf2
call _printf
```

```
    add esp, 8
    jmp .menu_loop

    .invalid_number_gcf:
    ; Display error message for invalid number and loop to the menu
    push choice_num
    call _printf
    add esp, 4
    jmp .menu_loop

.feature:
    ; Display message for invalid feature choice
    push choice_num
    call _printf
    add esp, 4

    ; Read a valid divisor
    push prompt_second_num
    call _printf
    add esp, 4

    ; Read and validate the new divisor
    push second_num  ; Store the new divisor
    push inputformat_numbers
    call _scanf
    add esp, 8

    mov eax, [second_num]
    cmp eax, 1
    jl .feature
    cmp ebx, 99
    jg .feature

    ; Perform GCF calculation with the valid numbers
    push dword [first_num]
    push dword [second_num]
    mov eax, [esp+4]
    mov ebx, [esp+8]

    ; Call GCF function
    call _calculate_gcf

    ; Display the result
    push eax
    push result_gcf2
    call _printf
    add esp, 8
    jmp .menu_loop
.done:
    ret
```

Pass-by-Reference

```
Insert your assembly (NASM) code here, be sure to add comments to
describe each line/set of codes...
;Learning Task (SAL Part 2 - Modular Programming) ||
pass-by-reference by group 12

section .data

  welcome_msg db " Hi! This is Group 12, and we are here to assist
you with LCD and GCF calculations for two-digit numbers (1 to 99)",
10, 0
  menu_msg db " === LCD and GCF CALCULATOR by Group 12 ===", 0xA
         db " [0] Exit", 0xA
         db " [1] LCD", 0xA
         db " [2] GCF", 10, 0
  user_choice_prompt db " Enter your choice: ", 0
  user_choice_format db "%d", 0
  lcd_msg db " === LCD ===", 10, 0
  gcf_msg db " === GCF === ", 10, 0
  first_num_prompt db " Enter First Number: ", 0
  first_num_format db "%d", 0
  second_num_prompt db " Enter Second Number: ", 0
  second_num_format db "%d", 0
  lcd_result_format db " LCD: %d", 10, 0
  gcf_result_format db " GCF: %d", 10, 0
  invalid_choice_msg db " Invalid choice. Please enter a valid
option.", 10, 0
  invalid_input_msg db " Input should be between 1 and 99. Please
enter a valid input.", 10, 0
  exit_msg db " Thank you! ", 0

; store the user's input
section .bss
  user_choice resd 1
  first_num resd 1
  second_num resd 1
  result resd 1

; section that contains the program logic
section .text

; Function to calculate the Greatest Common Divisor (GCD) using
Euclidean algorithm
gcd_function:
    push ebp
    mov ebp, esp

    ; Load values from memory addresses (pass by reference)
    mov eax, [first_num]   ; Load the value from the memory address
pointed by first_num
    mov ebx, [second_num]  ; Load the value from the memory address
pointed by second_num
```

```
gcd_loop:
    cmp ebx, 0
    je gcd_done
    xor edx, edx
    div ebx
    mov eax, ebx
    mov ebx, edx
    jmp gcd_loop

gcd_done:
    ; Store the result at the memory address pointed by result (pass by reference)
    mov [result], eax
    pop ebp
    ret

; Function to calculate LCD using GCD function
calculate_lcd:
    ; Call the GCD function (pass by reference)
    push dword [first_num]
    push dword [second_num]
    call gcd_function

    ; Retrieve results from GCD function
    pop ebx
    pop eax

    ; Calculate LCD using GCD result
    imul eax, ebx
    idiv dword [result]

    ; Display the result
    push eax
    push lcd_result_format
    call _printf
    add esp, 8

    ; Return to the main menu
    jmp display_menu

; Function to calculate GCF using GCD function
calculate_gcf:
    ; Call the GCD function (pass by reference)
    push dword [first_num]
    push dword [second_num]
    call gcd_function

    ; Retrieve GCD result
    push dword [result]

    ; Display the result
    push gcf_result_format
    call _printf
    add esp, 8

    ; Return to the main menu
    jmp display_menu

; Function to handle invalid user choices
invalid_choice_handler:
    ; Check if the user's choice is valid (0-2)
    cmp dword [user_choice], 0
    jl invalid_choice_handler
    cmp dword [user_choice], 2
    jl invalid_choice_handler

    ; Display an error message
    push invalid_choice_msg
    call _printf
    add esp, 4

    ; Return to the main menu
    jmp display_menu

global _main
extern _printf
extern _scanf

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

    ; Jump to the main menu
    jmp display_menu

; Function to display the main menu
display_menu:
    ; Display the main menu options
    push menu_msg
    call _printf
    add esp, 4

    ; Display a prompt and get the user's choice
    push user_choice_prompt
    call _printf
    add esp, 4

    ; Read user's choice and store in user_choice variable
    push user_choice
    push user_choice_format
    call _scanf
    add esp, 8

    ; Check user's choice and perform corresponding action
    cmp dword [user_choice], 0
    je exit_program
    cmp dword [user_choice], 1
    je input_lcd_first_num
    cmp dword [user_choice], 2
    je input_gcf_first_num

    ; Handle invalid choices
    jmp invalid_choice_handler

; Function to get the first number for LCD calculation
input_lcd_first_num:
    ; Display LCD message
    push lcd_msg
    call _printf
    add esp, 4

    ; Display prompt for the first number
    push first_num_prompt
    call _printf
    add esp, 4

    ; Read the first number and store in first_num variable
    push first_num
    push first_num_format
    call _scanf
    add esp, 8

    ; Check if the input is valid (1 to 99)
    cmp dword [first_num], 1
    jl invalid_lcd_input_first_num
    cmp dword [first_num], 99
    jg invalid_lcd_input_first_num

    ; Jump to input_lcd_second_num
    jmp input_lcd_second_num

invalid_lcd_input_first_num:
    ; Display error message for invalid input
    push invalid_input_msg
    call _printf
    add esp, 4

    ; Repeat the input_lcd_first_num section
    jmp input_lcd_first_num

; Function to get the second number for LCD calculation
input_lcd_second_num:
    ; Display prompt for the second number
    push second_num_prompt
    call _printf
    add esp, 4

    ; Read the second number and store in second_num variable
    push second_num
    push second_num_format
    call _scanf
    add esp, 8

    ; Check if the input is valid (1 to 99)
    cmp dword [second_num], 1
    jl invalid_lcd_input_second_num
    cmp dword [second_num], 99
    jg invalid_lcd_input_second_num
    ; Perform LCD calculation
    jmp calculate_lcd

invalid_lcd_input_second_num:
    ; Display error message for invalid input
    push invalid_input_msg
    call _printf
    add esp, 4

    ; Repeat the input_lcd_second_num section
    jmp input_lcd_second_num

; Function to get the first number for GCF calculation
input_gcf_first_num:
    ; Display GCF message
    push gcf_msg
    call _printf
    add esp, 4

    ; Display prompt for the first number
    push first_num_prompt
    call _printf
    add esp, 4

    ; Read the first number and store in first_num variable
    push first_num
    push first_num_format
    call _scanf
    add esp, 8

    ; Check if the input is valid (1 to 99)
    cmp dword [first_num], 1
    jl invalid_gcf_input_first_num
    cmp dword [first_num], 99
    jg invalid_gcf_input_first_num

    ; Jump to input_gcf_second_num
    jmp input_gcf_second_num

invalid_gcf_input_first_num:
    ; Display error message for invalid input
    push invalid_input_msg
    call _printf
    add esp, 4

    ; Repeat the input_gcf_first_num section
    jmp input_gcf_first_num

; Function to get the second number for GCF calculation
input_gcf_second_num:
    ; Display prompt for the second number
    push second_num_prompt
    call _printf
    add esp, 4

    ; Read the second number and store in second_num variable
    push second_num
    push second_num_format
    call _scanf
    add esp, 8

    ; Check if the input is valid (1 to 99)
    cmp dword [second_num], 1
    jl invalid_gcf_input_second_num
    cmp dword [second_num], 99
    jg invalid_gcf_input_second_num
    ; Perform GCF calculation
    jmp calculate_gcf

invalid_gcf_input_second_num:
    ; Display error message for invalid input
    push invalid_input_msg
    call _printf
    add esp, 4

    ; Repeat the input_gcf_second_num section
    jmp input_gcf_second_num

; Function to exit the program
exit_program:
    ; Display exit message
    push exit_msg
    call _printf

    ; Return from the main program
    ret
```

GROUP ASSIGNMENT

NAME OF MEMBER	TASK ACCOMPLISHED
John Peter Alcoy	**Documentation, Video Recording**
Roberto Bayos Jr.	**Documentation, Video Recording, Pass by Value code**
Marc Christian Tumaneng	**Documentation, Video Recording, Pass by Reference code**