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

COLLEGE *of* COMPUTER STUDIES

CS 318 – Architecture and Organization

LEARNING TASK (ARRAYS)

GROUP NO: 12 SECTION: BSCS 3A

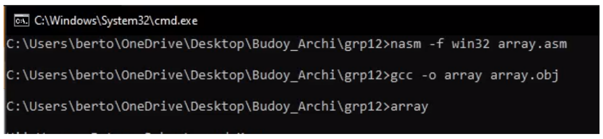
GROUP MEMBERS: Roberto Bayos Jr.
John Peter Alcoy
Marc Christian Tumaneng

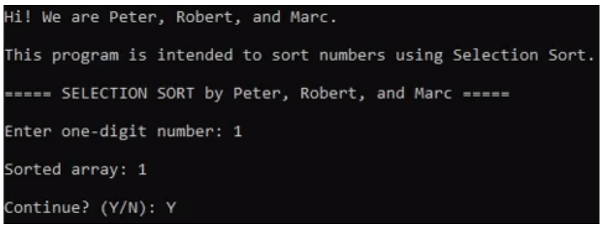
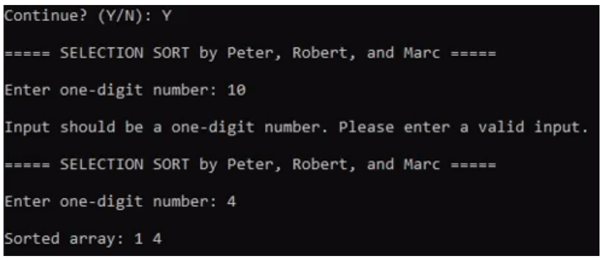
SAMPLE RUN

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

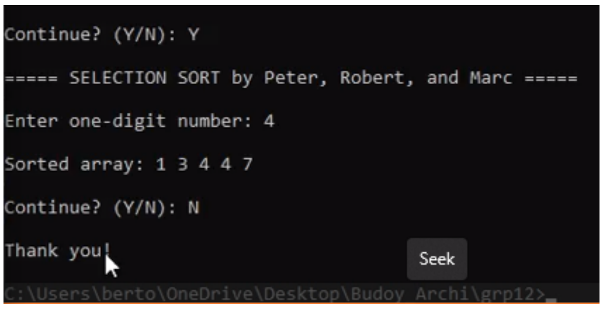
YouTube link: *https://youtu.be/jDlScuuZB00*

Step 1	*Image 1*

*As usual, Assemble the **"array.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 **"array.obj"** to get an executable program named **array**.*	
	C:\Users\berto\OneDrive\Desktop\Budoy_Archi\grp12>nasm -f win32 array.asm
	C:\Users\berto\OneDrive\Desktop\Budoy_Archi\grp12>gcc -o array array.obj
	C:\Users\berto\OneDrive\Desktop\Budoy_Archi\grp12>array

Step 2 *After running the program, the introductory message will be displayed along with the title and also the first prompt input where the user will input a one-digit number.*	*Image 2* Hi! We are Peter, Robert, and Marc. This program is intended to sort numbers using Selection Sort. ===== SELECTION SORT by Peter, Robert, and Marc ===== Enter one-digit number: 1 Sorted array: 1 Continue? (Y/N): Y
Step 3 *Now, the program will ask the user if he/she wants to continue or not, by typing **Y/N.** The user typed Y so the program continues, and enters a two digit number which is **"10",** and the program displays an error message since this program only reads one-digit number.*	*Image 3* Continue? (Y/N): Y ===== SELECTION SORT by Peter, Robert, and Marc ===== Enter one-digit number: 10 Input should be a one-digit number. Please enter a valid input. ===== SELECTION SORT by Peter, Robert, and Marc ===== Enter one-digit number: 4 Sorted array: 1 4

Step 3	*Image 3*

Again, same process type ***Y*** *to continue and put a one digit number, now we have typed 7, 3, 4 and repeat again the 4 now it displays again but still in sorted form.*	
	
	`Continue? (Y/N): Y`
	`===== SELECTION SORT by Peter, Robert, and Marc =====`
	`Enter one-digit number: 4`
	`Sorted array: 1 3 4 4 7`
	`Continue? (Y/N): N`
	`Thank you!`
	`C:\Users\berto\OneDrive\Desktop\Budoy Archi\grp12>_`

SELECTION SORT in ASSEMBLY

Discuss how selection sort works. Give an example of selection sort using your assembly program.

Selection sort works when the user types any number then displays it in order from small number to large number. but in our program case, it should accept only a one-digit number so yes from 1-9 and as you can see the final output we got ***1, 3, 4, 4, 7.***

Initialization: The algorithm splits the array into two sections: sorted and unsorted. The sorted region is initially empty, and the unsorted area is the entire array.

Find the Minimum: The algorithm searches the unsorted region for the smallest element.

Step 2's minimum element is switched with the first element in the unsorted area.

The boundary that separates the sorted and unsorted areas has been shifted one element to the opposite side, effectively expanding the sorted region.

Steps 2-4 continue until all of the array has been sorted.

PROGRAM CODE

```
;CS 318 – Architecture and Organization || LEARNING TASK (ARRAYS)

section .data
    prompt_welcome db "Hi! We are Peter, Robert, and Marc.", 10, 0
    prompt_welcome2 db "This program is intended to sort numbers using Selection Sort.", 10, 0
    prompt_title db "===== SELECTION SORT by Peter, Robert, and Marc =====", 10, 0
    prompt_input db "Enter one-digit number: ", 0
    instr db '%d', 0
    prompt_output dd "Sorted array: ", 0
    outstr dd '%d ', 0
    prompt_continue db "Continue? (Y/N): ", 0
    choice db '%s', 0
    exit_msg db "Thank you!", 10, 0
    error_msg db "Input should be a one-digit number. Please enter a valid input.", 10, 0
    inv_msg db "Please type Y/N only", 10, 0
    newline db 10, 0

    sort times 20 db 1
    sortednum times 200 db 0
    val dd 0
    val1 dd 0
    val2 dd 10

section .bss
    array resb 1  ; Assuming a maximum of 10 elements in the array
    user_choice resb 2
    sorted_values resb 100


section .text
```

```
    global _main
    extern _printf
    extern _scanf
    extern _exit

_main:

    push newline
    call _printf
    add esp, 4

    push prompt_welcome
    call _printf
    add esp, 4

    push newline
    call _printf
    add esp, 4

    push prompt_welcome2
    call _printf
    add esp, 4

main_loop:

    push newline
    call _printf
    add esp, 4

    push prompt_title
    call _printf
    add esp, 4

    push newline
    call _printf
    add esp, 4

    push prompt_input
    call _printf
    add esp, 4

    push array
    push instr
    call _scanf
    add esp, 8

    push newline
    call _printf
    add esp, 4

    cmp byte [array], 1
    je valid_input
    cmp byte [array], 2
    je valid_input
    cmp byte [array], 3
    je valid_input
    cmp byte [array], 4
    je valid_input
    cmp byte [array], 5
    je valid_input
    cmp byte [array], 6
    je valid_input
    cmp byte [array], 7
    je valid_input
    cmp byte [array], 8
    je valid_input
    cmp byte [array], 9
    je valid_input
    call msg_function

    jmp main_loop

    push newline
    call _printf
    add esp, 4

msg_function:
    push error_msg
    call _printf
    add esp, 4

    jmp main_loop

valid_input:
    push prompt_output
    call _printf
    add esp, 4

    jmp append

append:
     ; append an element to the array
     mov eax, [array]
     mov ebx, [val]
     mov [sort+ebx], eax
     mov ecx, ebx ; this is to increment or add 4 to the counter
     add ecx, 4
     mov [val], ecx
     mov edx, [val1] ; this is to increment or add 1 to counter1
     add edx, 1
     mov [val1], edx
     add esp, 8

selection_sort:
    mov ecx, [val1]
    dec ecx

    mov esi, 0

outer_loop:
    mov edi, esi
    inc edi
inner_loop:
    mov eax, [sort + esi * 4] ; num1[i]
    mov ebx, [sort + edi * 4] ; num1[j]

    cmp eax, ebx
    jng not_greater ; jump if not greater (unsigned comparison)

    ; swap num1[i] and num1[j]
    mov edx, eax
    mov [sort + esi * 4], ebx
    mov [sort + edi * 4], edx

not_greater:
    inc edi ; move to the next element in the inner loop
    cmp edi, ecx ; check if we reached the end of the array
    jl inner_loop ; jump to the inner loop if not

    inc esi ; move to the next element in the outer loop
    cmp esi, ecx ; check if we reached the end of the array
    jl outer_loop ; jump to the outer loop if not


print_array_loop:
    push ebp
    mov ebp, esp
    mov eax, [val1]
    mov ebx, sort ; point bx to first number
    mov ecx, 0    ; load 0

loop:

    ; store the value because external function like printf modify
the value
    push ebx
    push eax
    push ecx

    ; print the value stored on stack
    push dword [ebx]
    push outstr
    call _printf
    ; clear the stack
    add esp, 8

    ; restore these values
    pop ecx
    pop eax
    pop ebx

    ; increment the counter
    inc ecx
    ; add 4 bytes to ebx
    add ebx, 4
    ; compare value stored in ecx and eax
    cmp ecx, eax
    jne loop
    ; destroy the stack
    mov esp, ebp
    pop ebp

    push newline
    call _printf
    add esp, 4



continue_prompt:

    push newline
    call _printf
    add esp, 4

    push prompt_continue
    call _printf
    add esp, 4

    push user_choice
    push choice
    call _scanf
    add esp, 8

    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
    call error_function
    jmp continue_prompt

next_iteration:
    jmp main_loop



error_function:
    push inv_msg
    call _printf
    add esp, 4

    ret

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**
John Peter Alcoy	**coding, documentation, video recording**
Marc Christian Tumaneng	**coding, documentation, video recording**