ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT073-3-2-CSLLT

Computer Systems and Low-Level Techniques

APU2F2206CS(CYB), APD2F2206CS(CYB), APD2F2206CS(DF), APD2F2206CS, APDUF2206CS

HAND OUT DATE : 6 DECEMBER 2022

HAND IN DATE : 16 FEBRUARY 2023

WEIGHTAGE : 50%

INSTRUCTIONS TO CANDIDATES:

1. Submit your assignment at the administrative counter.
2. Students are advised to underpin their answers with the use of references (cited using the APA System of Referencing).
3. Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.
4. Cases of plagiarism will be penalized.
5. The assignment should be bound in an appropriate style (comb bound or stapled).
6. Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.
7. You must obtain 50% overall to pass this module.

Name : Yam Chen Xi

TP Number : TP061635

Table of Contents

1.0 Introduction 1
2.0 Research and Analysis 1
2.1 Importance of Assembly Language 1
2.2 Low Level Language in Cybersecurity and Forensic Fields 2
3.0 System Design: Flowchart 3
3.1 Main PROC 3
3.2 Procedure 4
3.3 Macro 9
4.0 System Screenshot 10
5.0 Source Code 15
6.0 Conclusion 22
7.0 Self-Reflection 22
References 23

1.0 Introduction

This assignment outlines the development of an assembly language cash register system for APU Cybersecurity Club, implemented using Turbo Assembler (TASM). The primary function is to enable users to select their preferred event and calculate the corresponding total fee, with discounts available to members of the club. The events are categorised into workshop, competition and activity. The system also accounts for limited workshop slots, enhancing its efficacy and ensuring the event is able to accommodate suitable number of participants. This assignment also encompasses research and analysis of assembly language, highlighting its relevance in cybersecurity and forensic industries.

2.0 Research and Analysis

2.1 Importance of Assembly Language

Assembly language is a variety of low-level programming language that uses mnemonics with the intention of being understandable by humans. To allow programmer to have direct communication and manipulation with certain hardware of computer, assembler is utilised to compile codes written in assembly language into machine language, which is complex for humans to decrypt as it only consists of binary characters. Assembly language enables the computer to understand the commands in machine language without any other third party interval (Fernando, 2022). Every assembler is created for a particular computer architecture, and it has a specific assembly language tailored to that assembler (Computer Hope, 2022). Other than direct manipulation over computer hardware, assembly language is able to handle critical tasks and provide processors access to individual instructions. Assembly language is able to simplify the completion of complicated tasks as it requires less memory and execution time. This is due to the processor only required to implement essential code instructions to complete specific task. Hence, time critical applications that written in assembly language can execute precisely in time (Pal, 2022).

2.2 Low Level Language in Cybersecurity and Forensic Fields

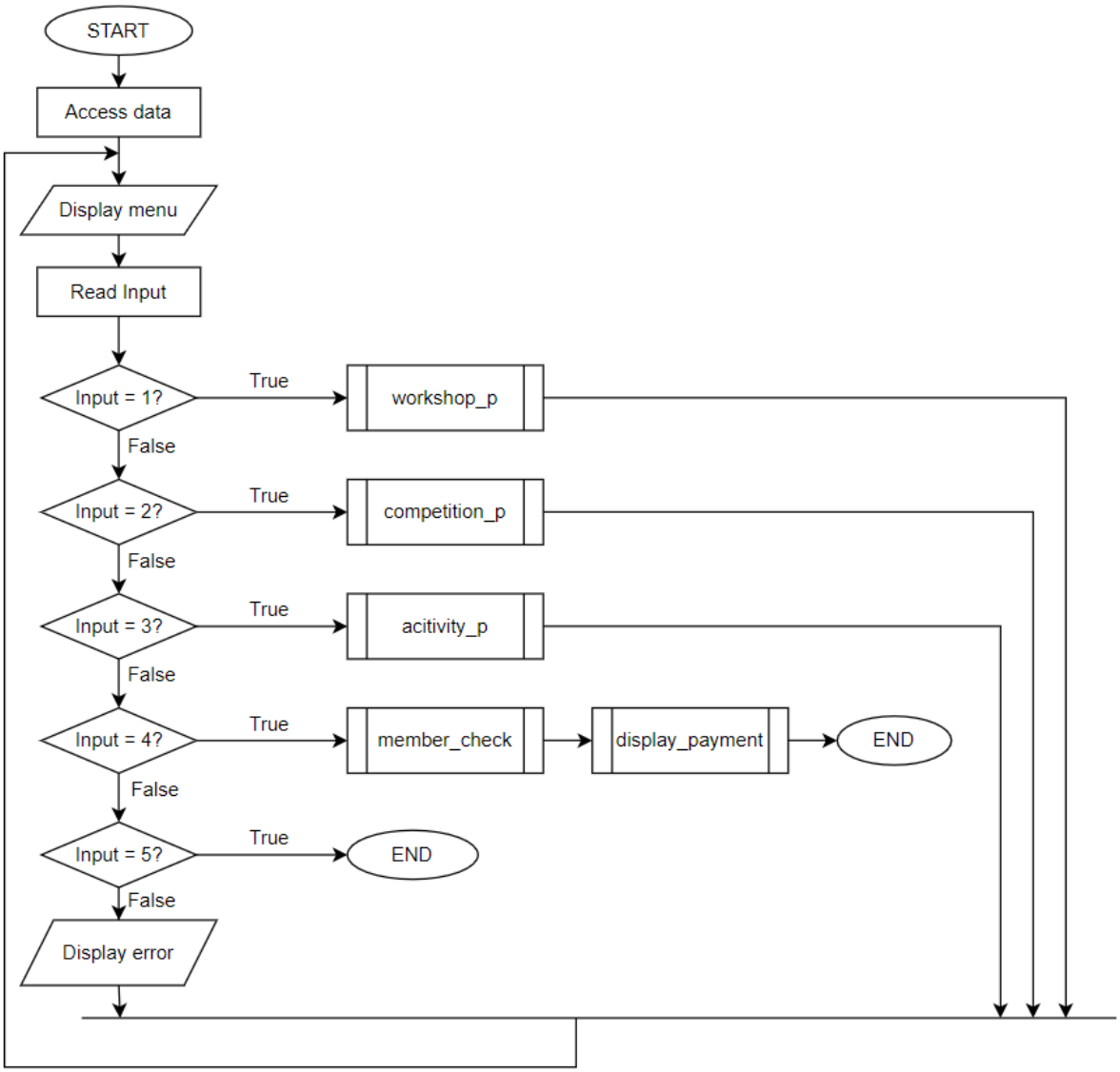
Low level language is commonly employed in cybersecurity and forensic industry due to the enhanced security and direct access to computer structures such as RAM and processors. Given the flaws that can be exploited at these low-level infrastructures, cybercriminals and hacker frequently search for penetration sites at this layer to use low level language to gain unauthorised access to the system (Sololearn, 2022). Cybercriminals are able to gain clear insights of systems and their memory management with a solid grasp of C or C++ language and proficiency with assembly language. Any high-level languages that are compiled into programs, such as C or C++ may be interpreted using Assembly language with the help of a debugger (Kool Stories, 2021).

Reverse engineering is a method in forensics enabling the disassemble of a subject framework in order to create depictions of the framework with a greater degree of assessment. Reverse engineering is able to examinate a program that does not contain any source code or proper documentation and trying to obtain information regarding the configuration and execution of the program (Shaid, 2014). With the first-hand exposure to the source code, it is simple to comprehend the function of the executable program. However, by learning assembly language or low-level language can be useful for malware analysis.

Malicious software also known as malware is commonly deployed across network intending to interfere a computer, server, or network by breaching the security and privacy of it. There are various types of malware attacks that can be found in the industry, such as ransomware, trojans, adware and more. Malware often hidden in email attachment, advertisement, false links and website. It can be triggered from simply one wrong click from a user and begin to run the malicious system that will damage the system (Artic Wolf, 2022). To secure a system to from further assaults, malware analysis is carried out to deciphers malware to discover the operation of it and examine infiltration of the system. Therefore, assembly language can be applied to reverse engineering to investigate malware (Puckett, 2022). For computerised forensic examination, low level programming languages are also used to search through binary data to extract ASCII and UNICODE characters. to easily identify concealed contents on either the whole document framework picture or unassigned storage in an easier way, the division of ASCII and UNICODE characters form the binary data can be carried out (Kumar et al., 2016).

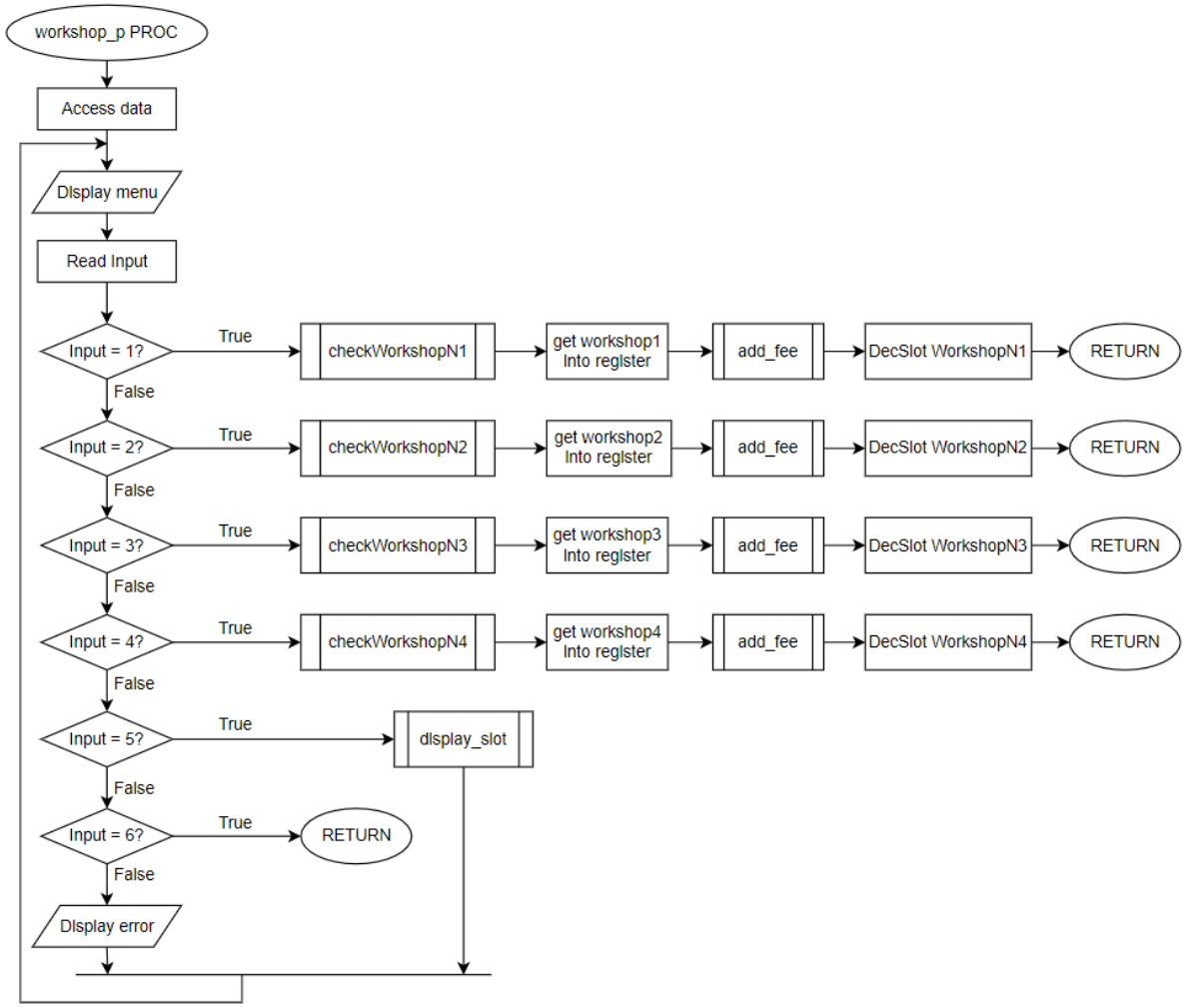
3.0 System Design: Flowchart

3.1 Main PROC

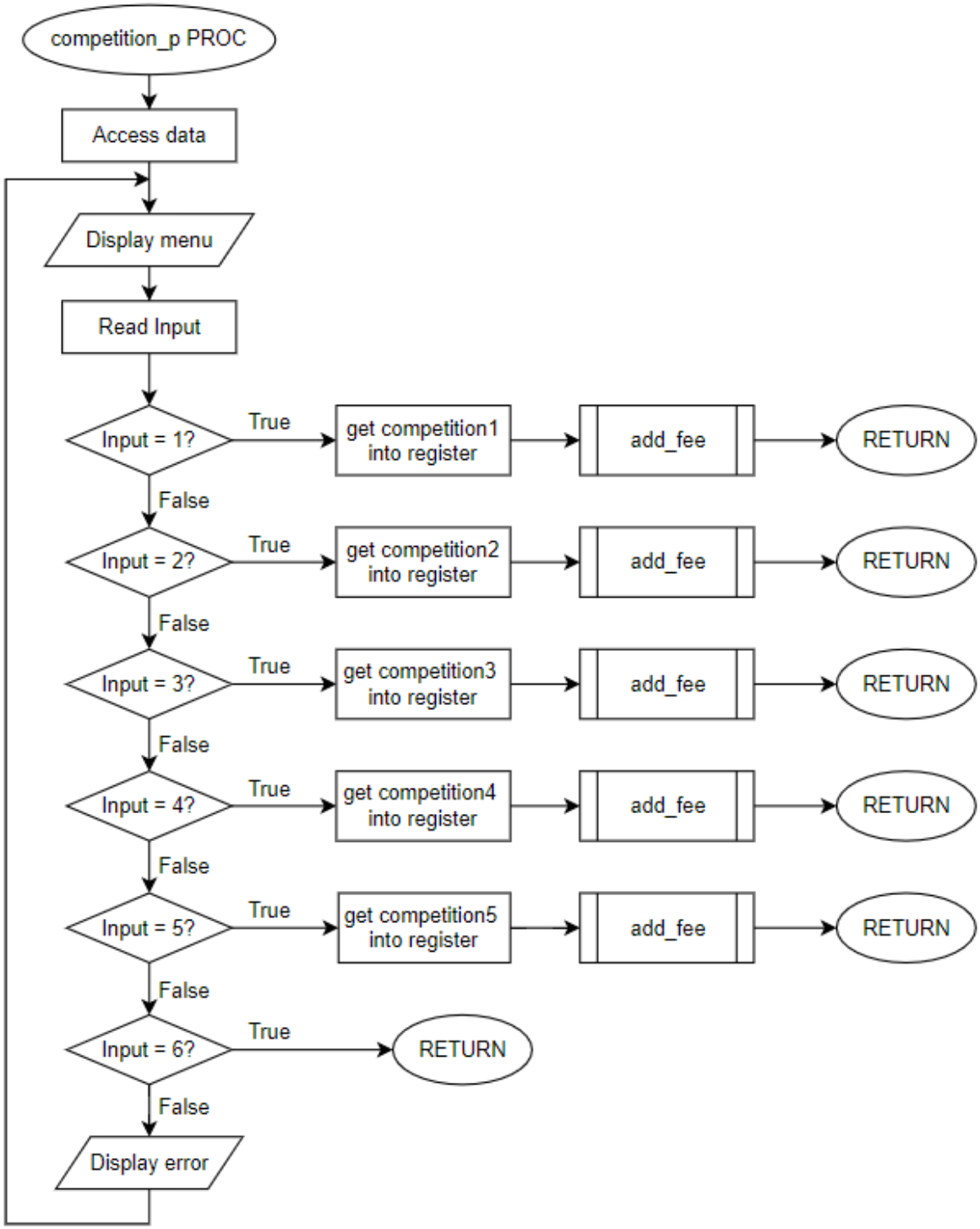


3.2 Procedure

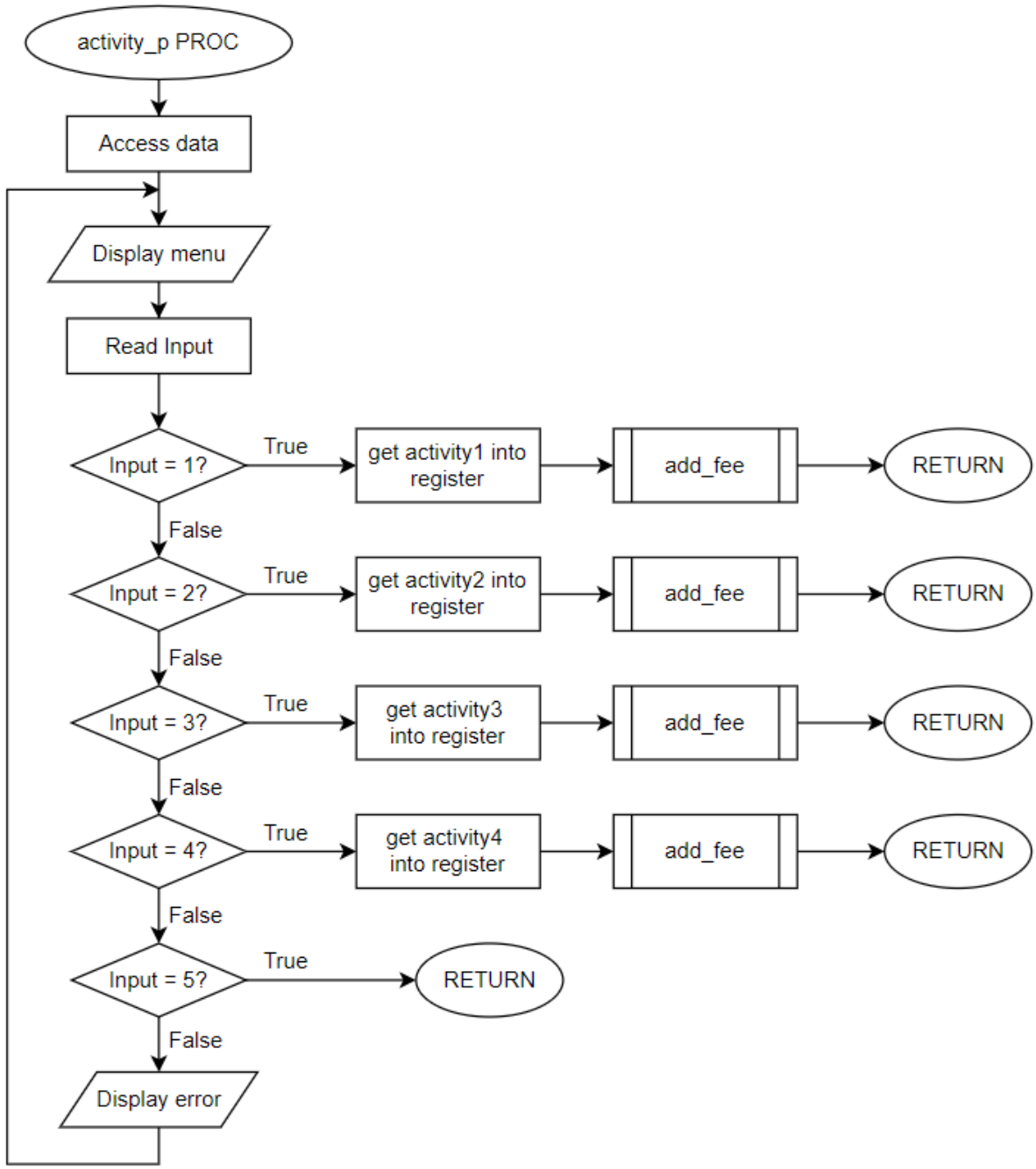
Workshop_p PROCC



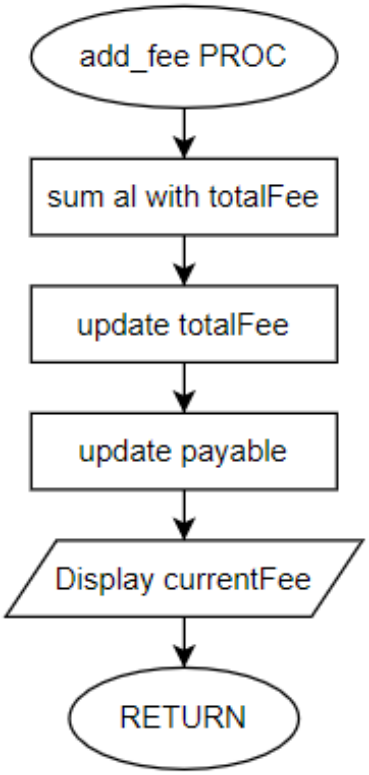
Competition_p PROC



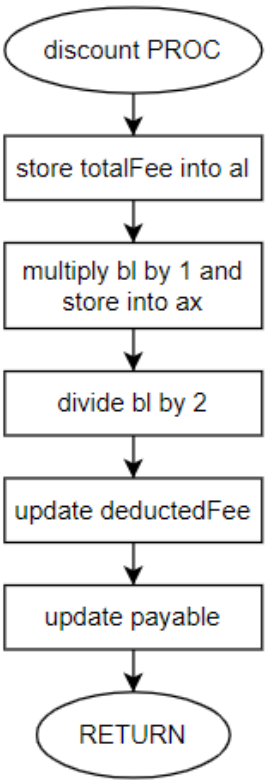
Activity_p PROC



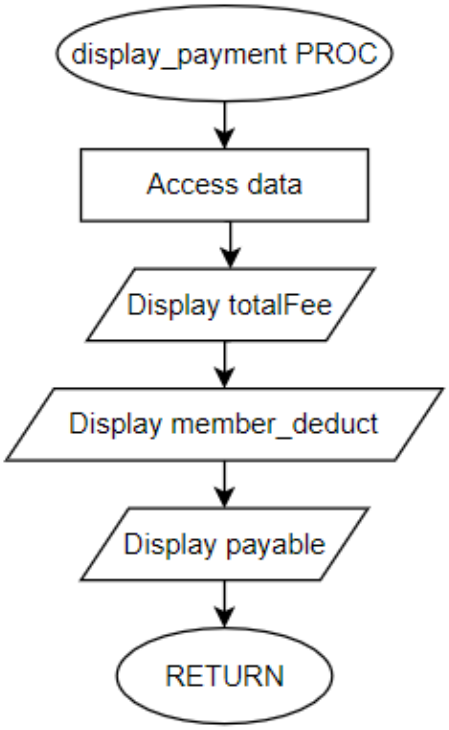
Add_Fee PROC



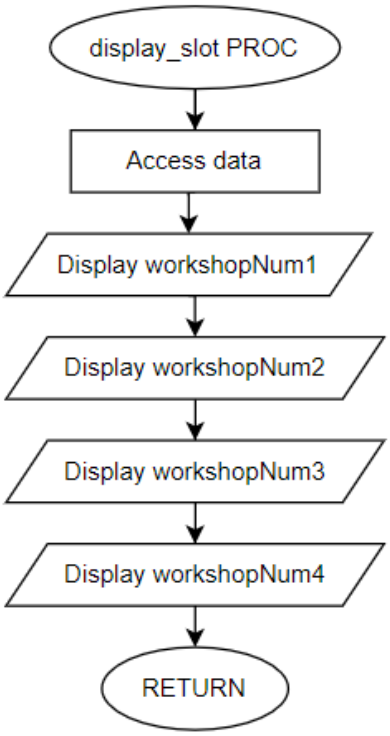
Discount PROC



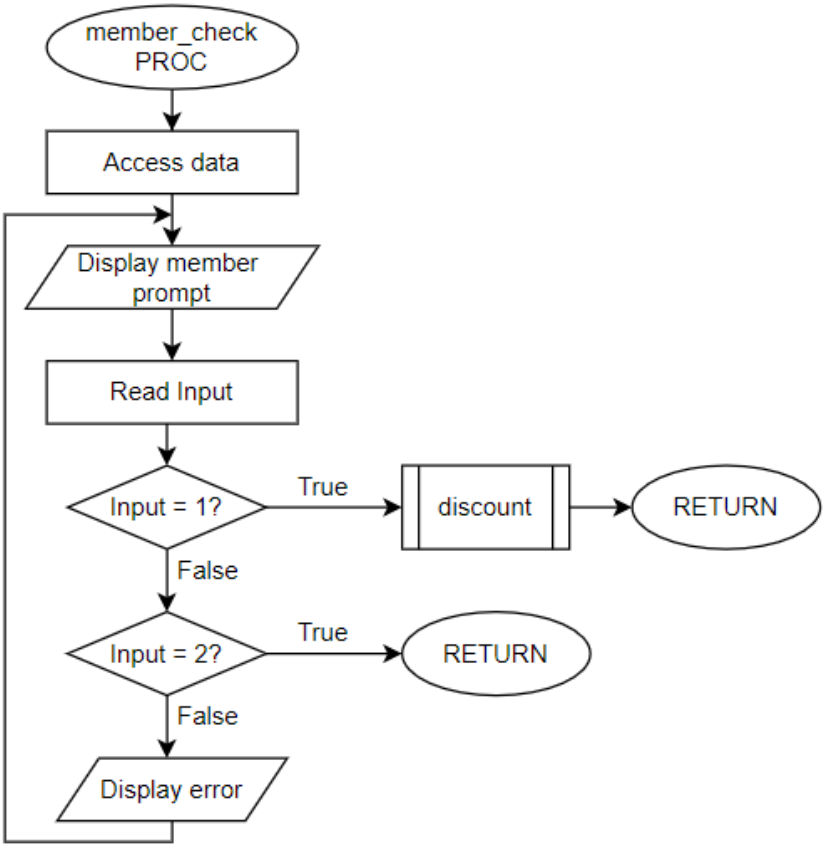
Display_Payment PROC



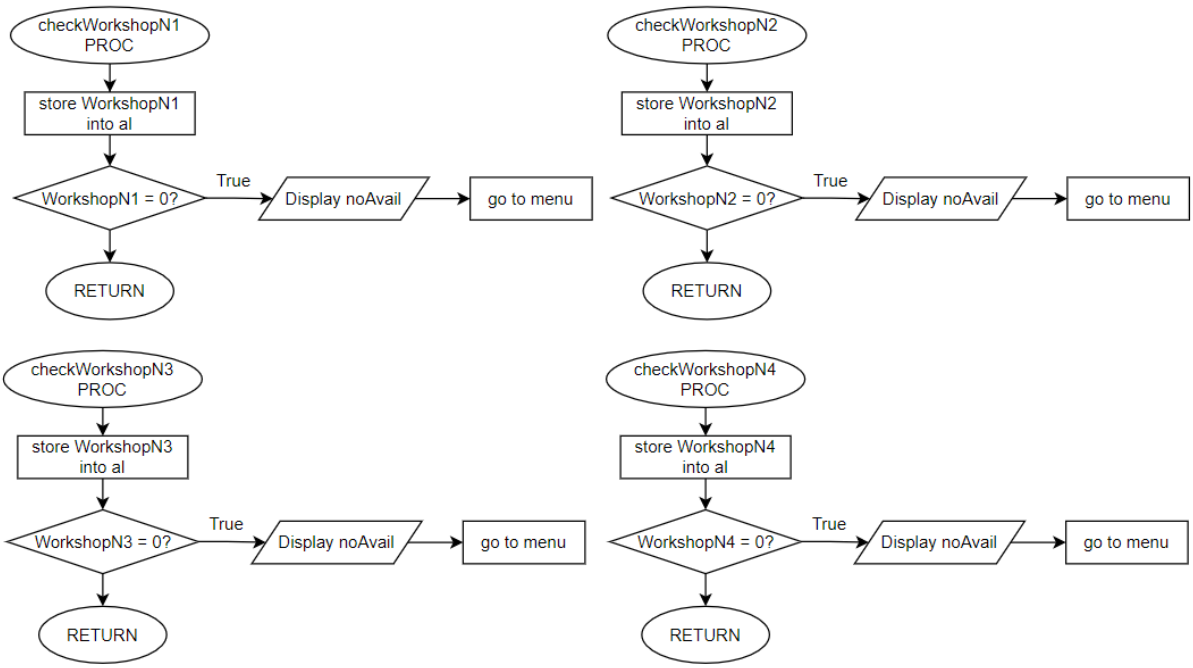
Display_Slot PROC



Member_Check PROC

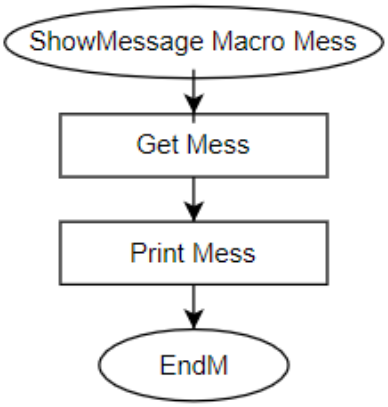


CheckWorkshopN PROC

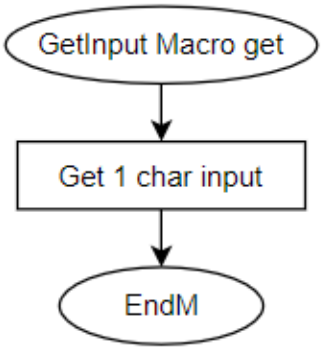


3.3 Macro

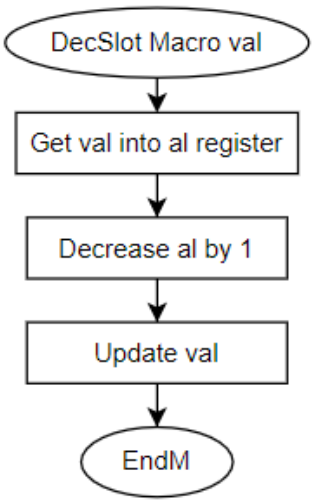
ShowMessage Macro Mess



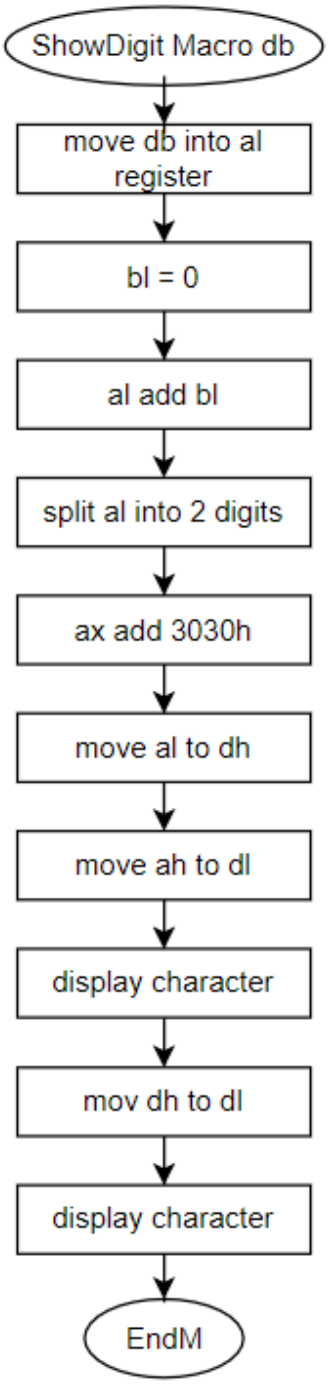
GetInput Macro get



DecSlot Macro val



ShowDigit Macro db



4.0 System Screenshot

```
C:\TASM>tp061635


==============================================
        FSEC-SS CASH REGISTER SYSTEM
==============================================
1. Workshop
2. Competition
3. Activity
4. Checkout
5. Quit
==============================================
Enter your choice: _
```

The system starts when the user running the program on TASM. The user is welcome with the main menu that contains 5 procedures which are workshop, competition, activity, checkout and quit. User are requested to enter their choice of procedure at the input section.

```
Enter your choice: 6
Invalid input. Please try again.
```

User input that are out of the range of 1 to 5 in main menu page, it will prompt out an error and required the user to input their choice again.

```
-----------------------------------------------
                Workshop
-----------------------------------------------
1. Phishing attacks workshop            RM10
2. Ethical Hacking Workshop             RM15
3. Cloud Security                       RM5
4. Cybersecurity Expo                   RM30
(5) Check Number of Available Slot
(6) Back
-----------------------------------------------
Enter your choice:
```

The user has chosen workshop, which is by inputting “1” in the main menu. It will lead the user to the workshop menu. This procedure consists of a list of workshops from option 1 to 4, option “5” which is provided for the user to check number of available slots for the workshops and option “6” which is return to main menu.

```
-----------------------------------------------
                Workshop
-----------------------------------------------
1. Phishing attacks workshop            RM10
2. Ethical Hacking Workshop             RM15
3. Cloud Security                       RM5
4. Cybersecurity Expo                   RM30
(5) Check Number of Available Slot
(6) Back
-----------------------------------------------
Enter your choice: 1

Added event into cart!
Current total: RM10
```

By inputting option within 1 to 4, it will add the selected workshop into cart and adding the price of the workshop with the charges of other selected events.

```
-----------------------------------------------
                Workshop
-----------------------------------------------
1. Phishing attacks workshop            RM10
2. Ethical Hacking Workshop             RM15
3. Cloud Security                       RM5
4. Cybersecurity Expo                   RM30
(5) Check Number of Participant
(6) Back
-----------------------------------------------
Enter your choice: 4
Event is fully registered!
```

If the selected workshop is fully registered and does not have any available slot anymore, the user will be notified that the selected event is unavailable to add into cart. The system will also not be adding the fee of the unavailable event into the cart.

```
Workshop Available Slot
- - - - - - - - - - - - - - - - - - - - - - -
1. Phishing attacks workshop    - 17
2. Ethical Hacking Workshop     - 05
3. Cloud Security               - 14
4. Cybersecurity Expo           - 00
```

When user input option “5” in the workshop page, it will prompt out the available slot for the workshop displayed in workshop menu. User can refer to this feature to select event they wanted to join with their friends.

```
-----------------------------------------------
                Competition
-----------------------------------------------
1. Doto CTF 2023                          RM13
2. System Hacking                         RM10
3. Cloud Defender                         RM15
4. Battle of Hacker                       RM10
5. Test IT Yourself                       FREE
(6) Back
-----------------------------------------------
Enter your choice: 1

Added event into cart!
Current total: RM23
```

The user has chosen competition, which is by inputting “2” in the main menu. The user will be directed to the competition menu. This procedure consists of a list of competition from option 1 to 5 and option “6” which is return to main menu. By inputting option within 1 to 5, it will add the selected competition into cart and sum with fee of others selected event.

```
-----------------------------------------------
                Activity
-----------------------------------------------
1. Digitalisation 2030                    FREE
2. Break the Internet                     RM10
3. Greener Digital World                  RM5
4. Security Awareness                     RM15
(5) Back
-----------------------------------------------
Enter your choice: 1

Added event into cart!
Current total: RM10
Event is free of charge!
```

The user has chosen activity, which is by inputting “3” in the main menu. The user will be guided to the activity menu. This procedure consists of a list of activity from option 1 to 4 and option “5” is used to return to main menu. By inputting option within 1 to 4, it will add the selected activity into cart and sum with fee of others selected event. However, if the event is free to participate, the user will not be charged for any payment for the specific event.

```
Are you a member?
1 - Yes  2 - No
Option: _
```

The user input 4 in the main menu, it will direct the user to checkout page. Starting by asking whether the user has membership. The user is required to input 1 for yes and 2 for no. Once the user input their option, the system will generate a receipt accordingly.

```
Are you a member?
1 - Yes  2 - No
Option: 1

        - - - - - - - - -
        Checkout
        - - - - - - - - -
Total Fee                : RM40
Member Discount (50%)    : RM20
Total Payable            : RM20

Thank you! Have a great day!
```

```
Are you a member?
1 - Yes  2 - No
Option: 2

        - - - - - - - - -
        Checkout
        - - - - - - - - -
Total Fee                : RM40
Member Discount (50%)    : RM00
Total Payable            : RM40

Thank you! Have a great day!
```

If the user is a member, the total fee will be deducted by member discount which is 50% off and generate the total payable. However, if the user is not a member, the total fee will not have any discount. Once the receipt is printed out, the system ends with a “Thank you” prompt.

5.0 Source Code

```
.data

; Main menu
    menuP       db 10,10,'==============================================',10
                                db 9,'FSEC-SS CASH REGISTER SYSTEM',10
                db '==============================================',10
                                db '1. Workshop',10
                db '2. Competition',10
                db '3. Activity',10
                db '4. Checkout',10
                                db '5. Quit',10
                db '==============================================$',10,0
```

```
; Input prompt
    option_prompt       db 'Enter your choice: $',10,0
    option_success      db 10,'Added event into cart!$'
    freeRegister        db 10,'Event is free of charge!$'
    memberP             db 10,'Are you a member?',10
                        db '1 - Yes',9,' 2 - No',10
                        db 'Option: $',0
    currentFee          db 10,'Current total: RM$'
```

The data above stores the message lines to be displayed in the system. Menu are structured in data section to be displayed as whole instead of separate line for each sentence. For input prompt however, the sentence will be displayed according to the code execution.

```
; Price List
    workshop1   db 10
    workshop2   db 15
    workshop3   db 5
    workshop4   db 30
    competition1        db 13
    competition2        db 10
    competition3        db 15
    competition4        db 10
    competition5        db 0
    activity1   db 0
    activity2   db 10
    activity3   db 5
    activity4   db 15

; Calculation
    totalFee    db 0
    deductFee   db 0
    payable     db 0

; Workshop available slot
    workshopN1  db 18
    workshopN2  db 5
    workshopN3  db 14
    workshopN4  db 0
```

The data above stores the integer value of the variable. It is used in the system to carry out calculations and availability checking.

```
ShowMessage Macro Mess
        mov ah, 09h     ;print message
        mov dx, offset Mess
        int 21h
EndM

; Get input from user
GetInput Macro get
        mov ah,1        ;read char
        int 21h
EndM
```

ShowMessage Macro Mess is used in the system to display message. Example, “ShowMessage menuP” can be found in the MAIN PROC to display the menu of the system.

```
ShowDigit Macro db
    mov al,db
        mov bl,0
        add al,bl

        aam
        add ax,3030h

        mov dh,al
        mov dl,ah

        mov ah,2
        int 21h

        mov dl,dh
        mov ah,2
        int 21h
EndM
```

ShowDigit Macro db is used in the system to display integer of db variable. Example, “ShowDigit workshop1” can be found in the workshop_p PROC to display number of available slots for workshop1.

```
DecSlot Macro val
        mov al,val
        dec al
        mov val, al
EndM
```

DecSlot Macro val can be found in workshop_p PROC to decrease the number of workshop available slot when the user has chosen to add event to cart.

```
MAIN PROC
        ;access database
        mov ax,@Data
        mov ds,ax

menu:
        ; display menu and ask user for input
        ShowMessage menuP
        ShowMessage CRLF
        ShowMessage option_prompt
        GetInput

        cmp al,49               ; comparing input to 49 Decimal which equals to 1
        jne competition         ; if input is not 1, jump to competition
        call workshop_p         ; if input is 1, call workshop_p
        jmp begin               ; loop menu

        competition:
                cmp al,50       ; comparing input to 50 Decimal which equals to 2
                jne activity
                call competition_p
                jmp begin
        activity:
                cmp al,51       ; comparing input to 51 Decimal which equals to 3
                jne checkout
                call activity_p
                jmp begin
        checkout:
                cmp al,52       ; comparing input to 52 Decimal which equals to 4
                jne quit
                call member_check
                call display_payment
                jmp end_main
        quit:
                cmp al,53       ; comparing input to 53 Decimal which equals to 5
                jne error1      ; if input not within 1 to 5, invalid input occur
                jmp end_main    ; end program

        error1:
                ShowMessage error

        begin:
                loop menu

        end_main:
                mov cx, 5
                ShowMessage CRLF
                ShowMessage thankyou

; End program
        mov ah,4ch
        int 21h

MAIN ENDP
```

The code above is the main procedure of the system. It will first display the menu and prompt for user input. The user input will be compared to continue to the allocated function. If the

input is neither in between 1 to 5, it will display error. The system will be in loop until the user has selected to checkout or quit the system.

```
; Program that manage workshop
workshop_p PROC
        mov ax,@Data
        mov ds,ax

workshop_menu:
        ; Display workshop list and ask user for input
        ShowMessage workshopP
        ShowMessage CRLF
        ShowMessage option_prompt
        GetInput

        cmp al,49               ; comparing input to 49 Decimal which equals to 1
        jne workshop2a          ; jump to workshop2, if input is not 1
        call checkWorkshopN1    ; call function to check event slot
        mov al,workshop1        ; get workshop1 db and store in al
        call add_fee            ; call function to add fee into total
        DecSlot workshopN1      ; deduct one slot for workshop1
        ret                     ; return to call

        workshop2a:
                cmp al,50       ; comparing input to 50 Decimal which equals to 2
                jne workshop3a
                call checkWorkshopN2
                mov al,workshop2
                call add_fee
                DecSlot workshopN2
                ret             ; return to call
```

workshop_p PROC is a procedure that manage workshop. In this system, there are two similar procedures that manage competition and activity, namely competition_p PROC and acitivity_p PROC. These procedures are used to display list of events and allow use to choose from it. The selected event will be added to the cart.

```
        checkSlot:
                cmp al,53
                jne exit_workshop
                call display_slot       ; call function to display number of available slot
                jmp workshop_menu       ; jump to workshop menu

        exit_workshop:
                cmp al,54
                jne error_workshop      ; if input not within 1 to 6, invalid input occur
                ret                     ; return to main menu

        error_workshop:
                ShowMessage error
                jmp workshop_menu       ; jump to workshop menu
workshop_p ENDP
```

As for workshop procedure, the user is provided feature to check availability of the workshop event by inputting “5” to call function to display slot available.

```
; Add fee into total fee
add_fee PROC
        add al, totalFee        ; sum db value and fee total fee
        mov totalFee, al        ; update total fee
        mov payable, al         ; update payable
        ; display current fee
        ShowMessage CRLF
        ShowMessage option_success
        ShowMessage currentFee
        ShowDigit totalFee
        ret
add_fee ENDP
```

This procedure sums the value in AL register with total fee and update total fee and payable by using the value in AL register. Then, it will display the current fee.

```
; Apply 50% discount on total fee
discount PROC
        mov al, totalFee        ; get total fee
        mov bl ,1               ; multiply by 1
        mul bl                  ; store into ax register

        mov bl, 2               ; divide by 2 for 50% discount
        div bl                  ; divide bl
        mov deductFee,al        ; update decucted fee
        mov payable,al          ; update payable

        ret
discount ENDP
```

Discount PROC is designed to apply 50% discount on total fee and update the deducted fee and payable.

```
; Check user workshop1 available slot
checkWorkshopN1 PROC
        mov al,workshopN1       ; get workshopN1 db value
        cmp al, 0               ; compare workshopN1 to 0

        je equalZero1           ; if equal to zero, jump to equalZero1
        jmp slotAvail1          ; if not equal to zero
        equalZero1:
                ShowMessage noAvail
                jmp menu        ; jump to main menu
        slotAvail1:
                ret             ; return to call
checkWorkshopN1 ENDP
```

To check workshop availability, checkWorkshopN1, checkWorkshopN2, checkWorkshopN3 and checkWorkshopN4 is used in the system. It will compare the workshopN db value to 0 and carry out the respective procedure. If workshopN is 0, it will prompt a message indicating the workshop is fully registered and go back to main menu. If the workshopN db value is not 0, it will return to call and continue the following procedure.

```
member_check PROC
        mov ax,@data
        mov ds,ax

        memberCheck:
                ; Display prompt and ask for input
                ShowMessage CRLF
                ShowMessage memberP
                GetInput

                cmp al,49       ; comparing input to 49 Decimal which equals to 1
                jne memberNo    ; if false, jump to memberNo
                call discount   ; call function to apply 50% discount on total fee
                ret             ; return to call

        memberNo:
                cmp al,50       ; comparing input to 50 Decimal which equals to 2
                jne error_member; jump to error_member, when input not within 1 and 2
                ret

        error_member:
                ShowMessage error
                jmp begin_member

        begin_member:
                loop memberCheck
member_check ENDP
```

To check membership, member_check PROC is used. This procedure is used to check whether the user is a member of APU cybersecurity club. If the user is a member, 50% discount will be applied on the total fee. Else, the user has to pay full payment for the selected event.

```
; Display checkout
display_payment PROC
        mov ax,@data
        mov ds,ax
        ShowMessage CRLF
        ShowMessage fee_prompt          ; display text from db
        ShowDigit totalFee              ; display integer val from db
        ShowMessage member_deduct
        ShowDigit deductFee
        ShowMessage payable_prompt
        ShowDigit payable
        ret
display_payment ENDP
```

Display_payment PROC is used in the system to display checkout receipt. It will display the total fee, member discount and total payable. After all these processes is completed, it will return to call to exit the system.

```
; Display workshop available slot
display_slot PROC
        mov ax,@data
        mov ds,ax
        ShowMessage CRLF
        ShowMessage workshopNum_prompt
        ShowMessage workshopNum1
        ShowDigit workshopN1
        ShowMessage workshopNum2
        ShowDigit workshopN2
        ShowMessage workshopNum3
        ShowDigit workshopN3
        ShowMessage workshopNum4
        ShowDigit workshopN4
        ShowMessage CRLF
        ret
display_slot ENDP
```

Display_slot PROC can be found in workshop_p PROC to display the available slots for the workshops.

6.0 Conclusion

Through the research on assembly language, it shows that assembly language is a powerful low-level programming language that leverages intuitive mnemonics to enable direct communication and manipulation of computer hardware. Its ability to provide enhanced security and direct access to computer structures make it a preferred choice in cybersecurity and forensic fields. Besides, the developed cash register system features a main menu that presents users with five procedures, including workshop, competition, activity, checkout, and quit. This efficient and user-friendly system is equipped with the necessary functionalities to support effective management of transactions.

7.0 Self-Reflection

It was challenging and rewarding to develop a cash register system using assembly language. I initially had a difficult time comprehending complexities of assembly language and how they relate to computer hardware. However, I was able to properly execute the system functions by learning and employing assembly language programming techniques. I had devoted a considerable amount of effort into studying about assembly language and how it was employed in the forensics and cybersecurity fields. This enabled me to comprehend the core ideas and guidelines needed to develop a system that is efficient. Furthermore, handling input validation, displaying the output data, and applying discounts and managing limited event slots were among the difficulties I ran through when designing the system. The great part of handling these difficulties has given me the chance to refine my problem-solving abilities and put definitive response into practise. Nevertheless, developing an assembly language cash register system had been a great experience. It gave me the opportunity to design a practical and efficient system while also strengthening my programming abilities and assembly language understanding.

References

Artic Wolf. (2022). *The 8 Most Common Types of Malware*. Arctic Wolf. https://arcticwolf.com/resources/blog/8-types-of-malware/

Computer Hope. (2022). *What is Assembly Language?* Computer Hope. https://www.computerhope.com/jargon/a/al.htm

Fernando, J. (2022). *Assembly Language*. Investopedia. https://www.investopedia.com/terms/a/assembly-language.asp

Kool Stories. (2021). *Cybersecurity- Languages to Learn*. Kool Stories. https://www.koolstories.com/blog/cybersecurity-languages-to-learn

Kumar, H., Radhakrishnan, L., Bin, F., Shah, F., Vannoshan, S., & Sundareson, A. (2016). An Analysis and Implementation Of Assembly Language Programming By Using TASM Incorporating With Security Concepts. *International Journal of Innovative Research in Science and Engineering*, *2*(11), 163–173. http://ijirse.com/wp-content/upload/2016/02/161ijirse.pdf

Pal, K. (2022). *Why is learning assembly language still important?* Techopedia. https://www.techopedia.com/why-is-learning-assembly-language-still-important/7/32268

Puckett, T. (2022). *Malware Analysis and Introduction to Assembly Language*. IBM Training and Skills Blog. https://www.ibm.com/blogs/ibm-training/malware-analysis-and-introduction-to-assembly-language/

Shaid, S. Z. M. (2014). *Introduction to Malware Reverse Engineering*. UTM Faculty of Computing. https://comp.utm.my/syed/files/2014/06/bookchapter-intro-to-malware-reverse-engineering.pdf

Sololearn. (2022). *What Do You Need To Learn For A Career In Cybersecurity?* Sololearn. https://www.sololearn.com/blog/what-do-you-need-to-learn-for-a-career-in-cybersecurity/