



ASSIGNMENT

CT073-3-2-CSLLT

COMPUTER SYSTEMS LOW LEVEL TECHNIQUES

APD2F2111CS(CYB)

HAND OUT DATE: 26 MAY 2022

HAND IN DATE: 3 August 2022

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

- 1 Assignment is to be submitted through online submission (Moodle).**
- 2 Students are advised to underpin their answers with the use of references (cited using the APA Referencing).**
- 3 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**
- 4 Cases of plagiarism will be penalized.**
- 5 You must obtain 50% overall to pass this module.**



INDIVIDUAL ASSIGNMENT

CT073-3-2-CSLLT

COMPUTER SYSTEMS LOW LEVEL TECHNIQUES

APD2F2111CS(CYB)

Summary of Assignment:

Module Code:	CT073-3-2-CSLLT
Module Title:	COMPUTER SYSTEMS LOW LEVEL TECHNIQUES
Student Name:	Li Jia Chong
Student TP:	TP056076
Weighting:	50%
Assessment Type:	Individual work
Hand out date:	26 May 2022
Submission date:	3 August 2022

Table of Contents

1.0 Introduction on Assembly Language	5
2.0 Research and Analysis.....	6
2.1. Contribution of Assembly Programming Language in Recent Applications	6
2.1.1 QEMU	6
2.1.2 MenuetOS	6
2.1.3 GHIDRA	6
2.1.4 Binary ninja	7
2.1.5 Radare2.....	7
2.1.6 Self-Justification	8
2.2 Significance of reverse engineering in image steganography	9
2.3 Describe usage of IDA Pro tool.....	10
3.0 Evaluation	13
3.1 low level programming language with any three high level programming languages ..	13
3.2 justify the importance of assembly language in cyber security/forensics.....	14
4.0 System Design using flowchart.....	15
5.0 System Screenshot.....	16
6.0 Limitation	20
7.0 Conclusion	21
8.0 Self Reflection.....	22
9.0 References.....	23

Figure 1 IDA VIEW	11
Figure 2 IDA Structure.....	11
Figure 3 Hex view in IDA	12
Figure 4 System Flow Chart.....	15
Figure 5 System menu page.....	16
Figure 6 Display left side triangle pattern	16
Figure 7 Display square pattern	16
Figure 8 Display ice cream pattern	17
Figure 9 Display triangle.....	17
Figure 10 Display diamond shape.....	18
Figure 11 Display Bye message.....	18
Figure 12 error message when insert invalid input.....	19

1.0 Introduction on Assembly Language

The field of programming is enormous and intricate. There are many distinct languages, and each one has a special style of resolving issues. For instance, a developer may use python to design a system and HTML to make a website. Assembly language is a low-level programming language that aids in computer communication or translating programming languages into machine code. Every computer or machine has an assembler that aids in turning the assembly code into an executable machine code. This language helps transform developer instructions into machine language for additional processing while simultaneously providing instructions that developers can comprehend. A programmer can write code that is human readable by using assembly language. If a developer has to create a program that requires a compiler, they will need to have understanding of the processor. Learning assembly language will assist a developer grasp the functionality of the machine's processor and memory. An assembly language programmer will be used to read registers and get pointers' and values' memory addresses. Assembly language will be used to track the tasks that a computer will process since machine language, which is essentially a sequence of numbers, is difficult to grasp. (Padamkar, n.d.)

In order to provide a new design for the digital banners for the events, I will be utilizing assembly language to create a suitable design form for the APU event management unit. Not only that, but I'll also use TASM to make five of the shapes for the event.

2.0 Research and Analysis

2.1. Contribution of Assembly Programming Language in Recent Applications

2.1.1 QEMU

QEMU was the current application to contribute assembly programming. A free and open-source emulator is QEMU. It will simulate the machine's processor, go through dynamic binary translation, and offer the machine with a range of hardware and device types, as well as making it possible for it to run several different guest operating systems. According to the research, this virtual system was developed to take the place of an outdated hardware-based system for teaching assembly language. (researchgate, 2015) The create virtual system incorporates all the development tools to give basic or advanced lessons on programming in contemporary assembly language. In the QEMU it had provide some of operating modes such as system emulation. These modes allow QEMU to run a variety of guest operating systems, including some popular ones like Linux, Windows, and Solaris. These modes also enable mimicking a number of instructions sets like x86, MIPS, 32-bit Arv7, and Micro Blaze. (pcmag, n.d.) .

2.1.2 MenuetOS

A PC operating system called MenuetOS was developed using x86 assembly programming, and it would only be written in assembly language (64bit and 32bit). Pre-emptive and real-time multitasking with support for multiple processors and a graphical user interface are features that are included in Menuet OS (graphical user interface). Menuet is not based on other operating systems like UNIX or the historically significant POSIX standards. The goal of the MENUET design was to get rid of the unnecessary layers that often exist between different Operating System components and cause programming problems and issues. Since the header may be created with essentially any other programming language, the MENUET application does not specialize in reserved assembly programming. However, the entire application is created for 64/32-bit assembly programming, making it simple to use this operating system using assembly. (MenuetOS, n.d.)

2.1.3 GHIDRA

Assembling malicious software may be done via reverse engineering techniques, as is well known. A computer application called a disassembler can assist in converting machine

language into assembly language. As a result, the output will frequently be changed in Disassembly to improve human readability.

The most recent program to contribute to assembly language is GHIDRA. GHIDRA is a versatile tool that the cybersecurity industry may use for free to aid in the analysis of malware. The NSA created GHIDRA, a reverse engineering tool that was already released in 2019 and has since shown to be well-liked by malware experts. Because the disassembly tools make it possible for a malware analyst to examine a malware sample's functioning without executing it, they are extremely helpful for the analysts and allow them to examine the virus's source code and determine its overall purpose. The debugger x64dbg will execute the malware as you browse around and examine the code, which is one of the distinctions between GHIDRA and x64dbg. Therefore, a malware analyst utilizing x64dbg must perform all functions, encrypt all files, and encrypt all files on the system they are using to analyze malware. In contrast to GHIDRA, however, it does not execute the code; instead, it maps out the malware's assembly code and enables the user to go forward and back through it without having an impact on the analysis device's filesystem. In light of this, GHIDRA is a better tool than x64dbg for malware analysts to use in detecting and mapping out malicious functionality. (Fox, 2022)

2.1.4 Binary ninja

Assembly language plays a significant part in reverse engineering, which is used to identify malware through disassembly, as is well known. The most current program to use assembly language is binary ninja. This is due to the fact that Vector35 INC Binary .s Ninja is a platform for reverse engineering. It may be utilized to decompile a binary and show the decompilation in liner or graph format. The automatic in-depth examination of the code will be displayed since machine language is symbolic and numerical, and it will also be producing data that aids in the understanding of the binary number. Because it will automatically carry out the different analyses from the binary, such as system function detection, cross-references for code, and value-set analysis, Binary Ninja has long been used for core analysis. The CPU architectures that Binary Ninja supports include 6502, ARMv7, MIPS, and x86(32/64bits) assembly language. (BINARYNINJA, 2016) (Noerenberg, 2018)

2.1.5 Radare2

Radare2(r2) is the final program to contribute assembly language. Radare2 is a framework for reverse engineering that analyses binaries and is made up of small tools that

may be used jointly or separately from the command line. For instance, it will develop around a disassembler for software that extracts the source code for assembly language from machine executables. The radare2's objective is to offer a user-friendly hexadecimal editor interface. It will also enable 64-bit offset to aid in data recovery from the system hard drive. The goal of this job is forensic or cyber security. The radare2's ability to conduct statistical analysis is one of its characteristics. For instance, it may assemble and disassemble software programs, focusing mostly on executables, and then display binary diffing with graphs and associated information such as symbol relocations and different types of data. Redare2's inclusion of the crucial characteristic of software exploitation. Redare2 may be used to check for malware and system vulnerabilities since it is a disassembler and low-level debugger. The program has capabilities that help cyber security professionals stop exploit development, such mitigation and detection. (Baldawa, 2020) (Azu, 2020)

2.1.6 Self-Justification

Based on the research , learning assembly language is very important not matter is in cyber security field or software developer field. It is because assembly language it helps to manipulate on the hardware directly, address critical issues concerning performance and also provide via to special instruction for processors. Based on the 5 example that contribute of assembly language which is QEMU, Binary ninja, Radare2, GHIDRA disassembler, Menuet OS. It shows that, assembly language still in use for Operating system and assembly language also have been typically to use on reverse engineering disassembler tools. Those evidence prove that assembly language very useful on hardware or reverse engineering.

2.2 Significance of reverse engineering in image steganography

There is the significance of using reverse engineering in image steganography because it will help decompiling and disassembling of executable files. As we know, image steganography is method which will hiding a secret data or image or the message into an object that is image file such as JPEG, TIFF, PNG, BMP and etc. In image steganography, the image will look like not difference between original image and the image that have been changed to steganography, it can't view through human eyes. (Ayush Kejariwal, 2019) Until now, having most of the application is provide image steganography, it is because digital marketing will be use image steganography to protect data transmission and in the numerous fields. By using reverse engineering, we are able to identify the message encryption process and concealing type for the encrypted message. For example, encrypted the message in one type of encryption method and using bytes for storing the bytes information hide into the Red Green Blue(RGB) portion of the pixel in the image. The RGB option which able user to insert their offset value .So the hidden data will perform in nonlinear type at different positions, it will be based on the RGB option or encryption through the reversing process it will help hiding messages in non-linear order for each pixel in the image. (Granville, 2017) So that, some of the big companies will be using steganography to make their high confidential data and encoded or encrypt with steganography techniques. The reason will be use reverse engineering in image steganography because we are able to use reverse engineering tools such as IDA pro to identify hidden algorithms reversely. (Lee, 2020)

2.3 Describe usage of IDA Pro tool.

A common analysis tool in the software security sector, IDA Pro is part of the toolkit of a software analyst, binary reverse engineer, or malware analyst. IDA Pro may be described as a disassembler that aids in exploring binary programs for the lack of available sample code and creating maps of their operation. The disassembler will display the instruction in the form of an assembly language symbol that will be carried out by the processor. Because assembly language is challenging to grasp, employing the technique built into IDA pro will make the code easier to read. The majority of cybersecurity professionals will utilize IDA pro as the root to decipher infections. IDA Pro may be used to analyze viruses as well as some obfuscated viruses. This is due to the fact that the IDA Pro debugger offers a static analysis capability of the disassembler that may be used to quickly navigate through the harmful code under investigation. The debugger will proceed directly through the phase and skip the obfuscation section, allowing it to acquire the detailed data. Due to its capacity to explore or evaluate vulnerabilities, IDA Pro has long been used in the cybersecurity industry. For instance, IDA Pro may be used to look into the causes of software errors and cyberattacks that happen because a system has a flaw that makes it simpler for the attacker to launch their assault. By looking into the issue, we can determine where the software vulnerability is and how to quickly address it. Finally, the majority of cybersecurity professionals will do COTS validation using the IDA pro tool. This is due to the fact that the majority of the software in use comes from other countries. Because certain source code makes the audit and rebuild functions useless, the software is difficult to understand. However, IDA Pro makes it simple for someone to determine if the program is carrying out its stated purpose. (e-spincorp, n.d.) When open dll file in IDA tools, it will show the IDA view, IDA structure, Hex view output, same as the figure below, which is figure1, figure2, figure3.



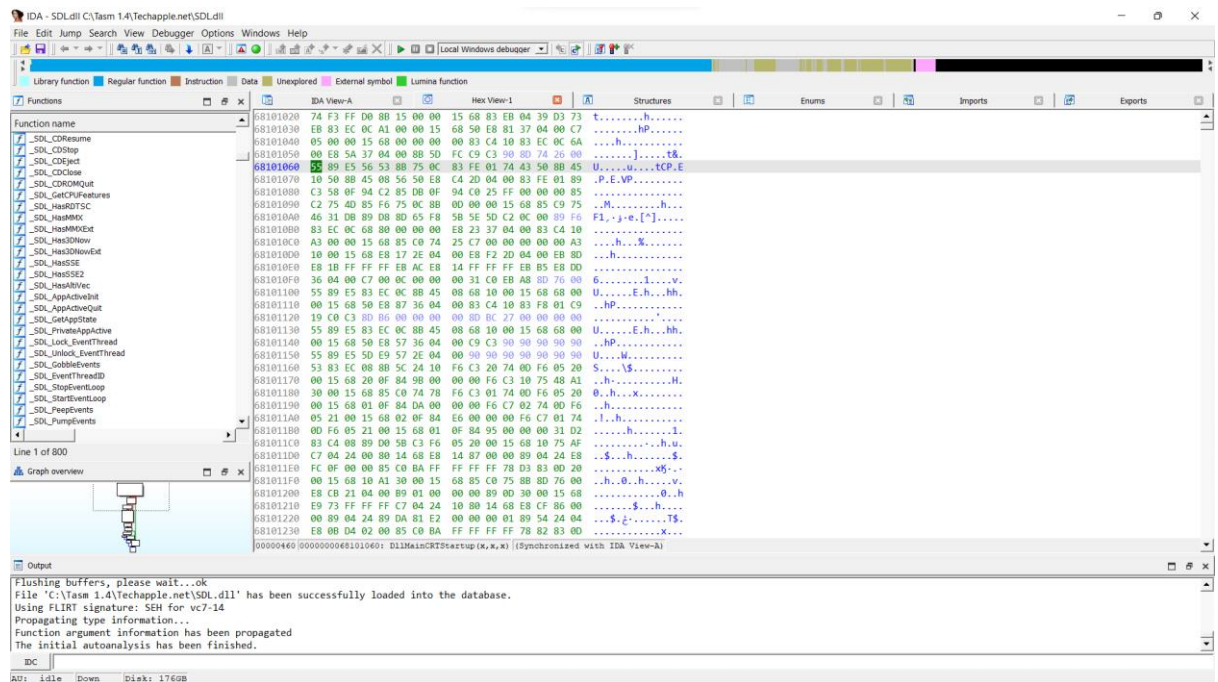


Figure 3 Hex view in IDA

3.0 Evaluation

3.1 low level programming language with any three high level programming languages

High-level programming and low-level programming are the two types of programming languages. Low level programming language that enables human and machine communication. The developer's code was able to create a set of instructions that the computer could follow. Assembly language and machine language are examples of low-level programming languages. A programmer will be able to build a form of programmer that is independent of the type of machine it is executed on because to high-level programming languages. Python, C, and Java are examples of high-level programming languages. (MKS075, 2020)

When compared to python, c, and java, assembly language has several distinctions. Comparing assembly language to python, c, and java, it is more challenging. It's because a program written in assembly language will employ an alphanumeric mnemonic code for a specific set of instructions, but programs written in python, c, and java can utilize certain English phrases and well-known mathematical symbols. For example, compare to c language and assembly language, c language provides portability which meaning it doesn't depend on the platform that use to cod, and it can be reuse in another coding platform, but assembly language doesn't provide any specific platform and the code is not portability, because the assembly language will be based on the processor architecture. Not only that, but in python, c, and java they need a compiler/interpreter for translation into machine code. Assembly language will need an assembler for translating the instructions of the machine language. Assembly language has higher memory efficiency than high level programming languages. Because assembly language has a high memory efficiency compared to high level programming, which has a relatively poor memory efficiency, they are able to consume less energy. The execution of assembly language is also quicker than that of python, c, and java since those programming languages need translation programs whereas assembly language does not. Furthermore, debugging in high level programming languages such as python, C, and java is quicker than in low level programming since low level languages are machine dependent and hard to define. There is some difference on type of processor between python, c and java compared to low level programming. For example, the type of processor that low level language use is the program that user coded is only for the processor in an assembly language will not run successful on any other processor. But python, java and c they are the language that easily run on any processor. (byjus, n.d.) (GeeksforGeeks , 2021) (Aticleworld, n.d.)

3.2 justify the importance of assembly language in cyber security/forensics.

The relevance of assembly language in the field of cyber security may be attributed to the fact that knowing it makes it easier to test software, identify security flaws, and identify malicious code. The prevention of malware attacks, for instance, would be aided by understanding assembly language. (Shaid, 2014) As is common knowledge, a malware attack is a cyberattack that originates from malicious software that causes the system to carry out unlawful operations. (Rapid7, n.d.) In order to define what the malware does, certain members of the cybersecurity team will collect samples of the virus and translate it into assembly code. It's because when an attacker creates malware, they usually utilize confusing code to prevent others from being able to read or comprehend the malware code and predict how it will operate. When malware sample code is discovered, we may use a tool like IDA Pro to convert the program into assembly language. For instance, if the program is run in machine language and machine code and assembly language are related, we may also utilize tools like a disassembler to translate machine language into assembly. After translating the malware code to assembly language, if we are experts in assembly code, we can comprehend what the program does and know how to stop it from spreading or detect it on a system. Finding indications of compromise (IOCs) is another method. IOC will refer to an occurrence that may be discovered during an inquiry, such as the discovery of a program-specific virus. We may expand the characteristic to fully understand the situation by applying the indications. Some of the IOCs will be the program's required postings to file-based IP addresses. So, by learning the assembly language will help to prevent malware attack such as ransomware, spyware and etc. (packetlabs, 2022)

4.0 System Design using flowchart

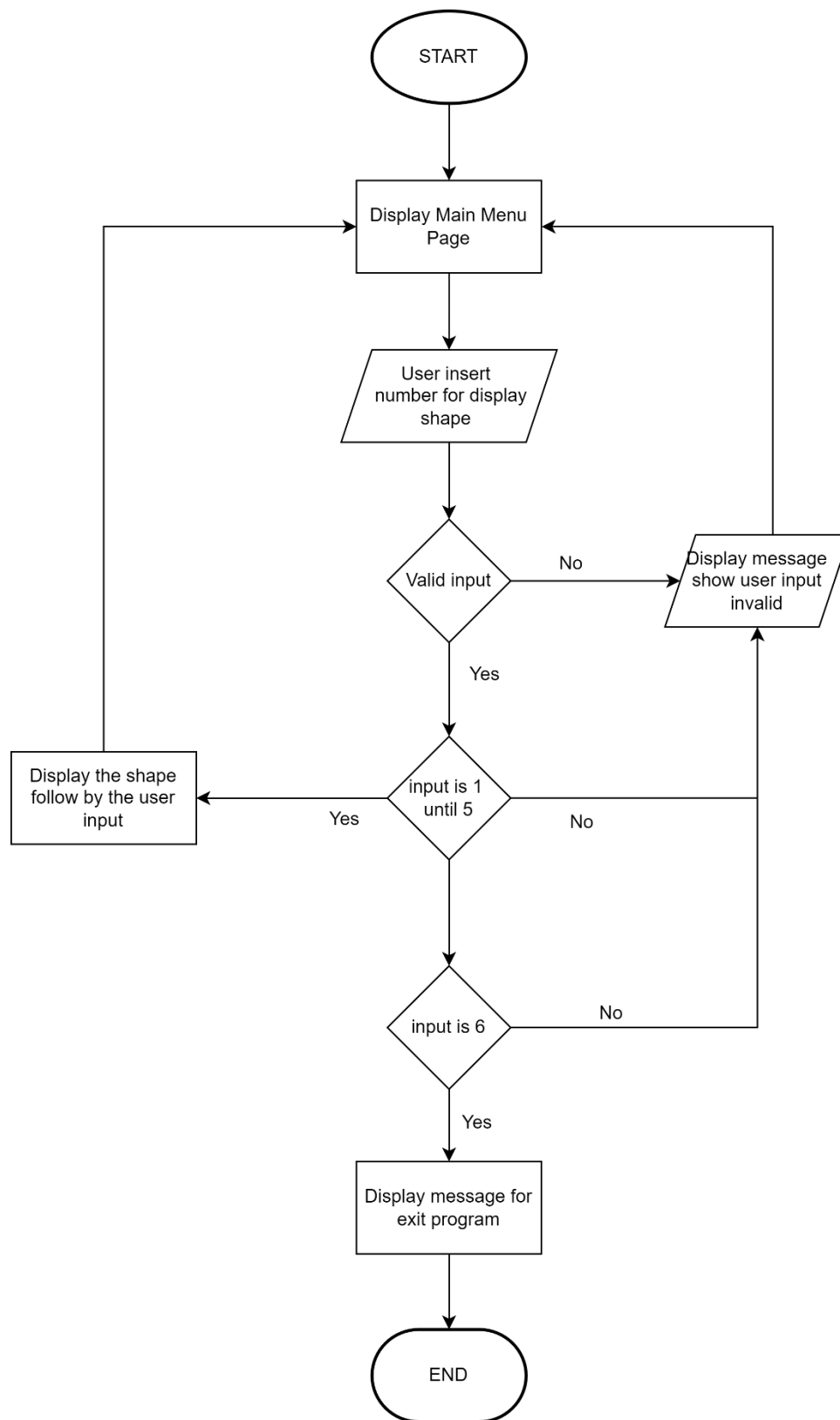


Figure 4 System Flow Chart

5.0 System Screenshot

```
      Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :_
```

Figure 5 System menu page

When the system had start, user able to saw the Menu page, which follow by the number for each of the shape. So user able to input the number to display the shape they want.

```
GUI Turbo Assembler x64
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :1

1
22
333
4444
55555
666666
7777777
88888888

      Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :
```

Figure 6 Display left side triangle pattern

When the user input number 1, the system will show the left side triangle, and the triangle will display follow by the number from 1 to 8.

```
GUI Turbo Assembler x64

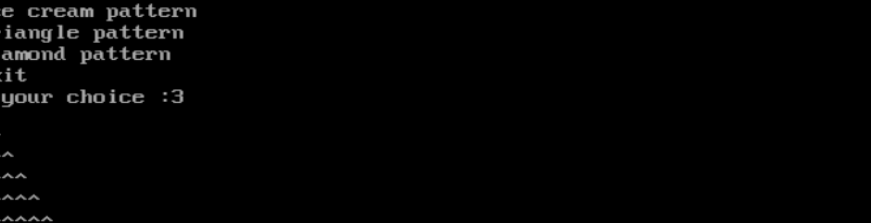
      Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :2

#####
#####
#####
#####
#####
#####

      Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :_
```

Figure 7 Display square pattern

When the user input number 2, the system will print out the square shape with the symbol ‘*’. Which having some spacing.



```
GUI Turbo Assembler x64
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :3

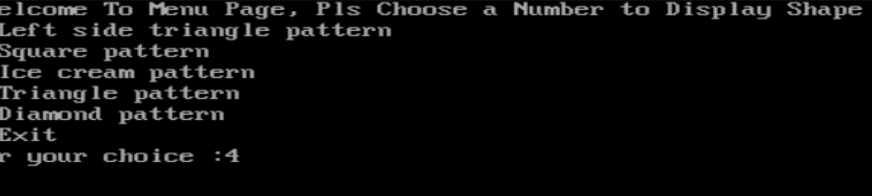
      ^
     ^^
    ^^^
   ^^^^
  ^^^^^
 ^^^^^^
^^^^^^

uuuuuuuu
uuuuuu
uuuuu
uuuu
uuu
u

Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :_
```

Figure 8 Display ice cream pattern

When user input number 3, the system will print out the ice cream shape, the shape will display with '^' and 'v'.



```
GUI Turbo Assembler x64
Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :4

  *
 ***
*****
*****
*****
*****
*****
*****

Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :
```

Figure 9 Display triangle

When user input number 4, the system will print out the triangle shape, the shape will display with symbol ‘.’

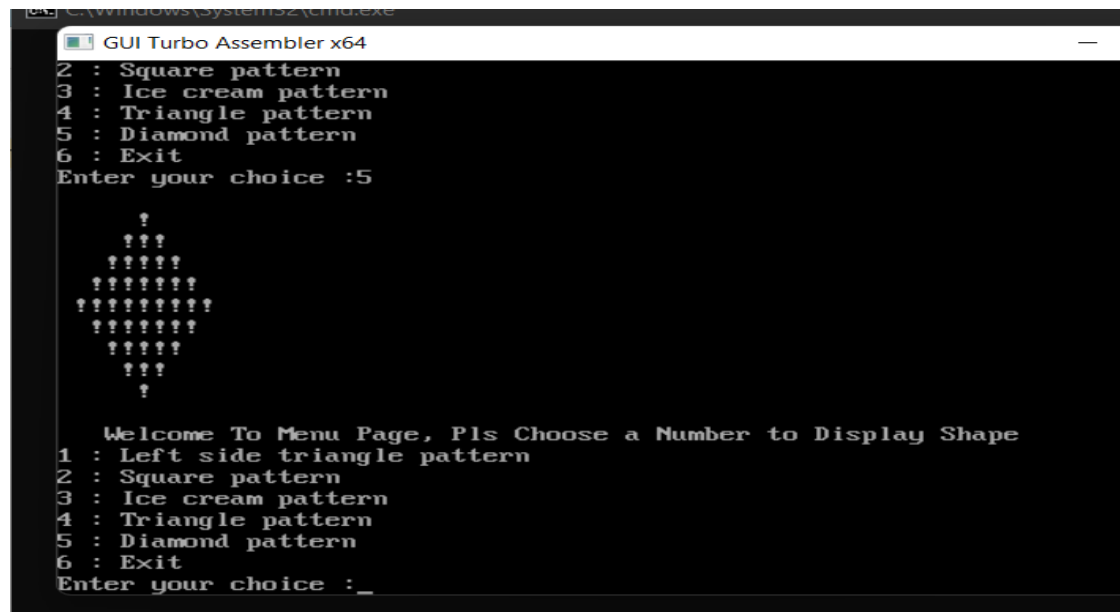


Figure 10 Display diamond shape

Lastly, when user input number 5, the diamond shape will be show, it will display with ‘!’

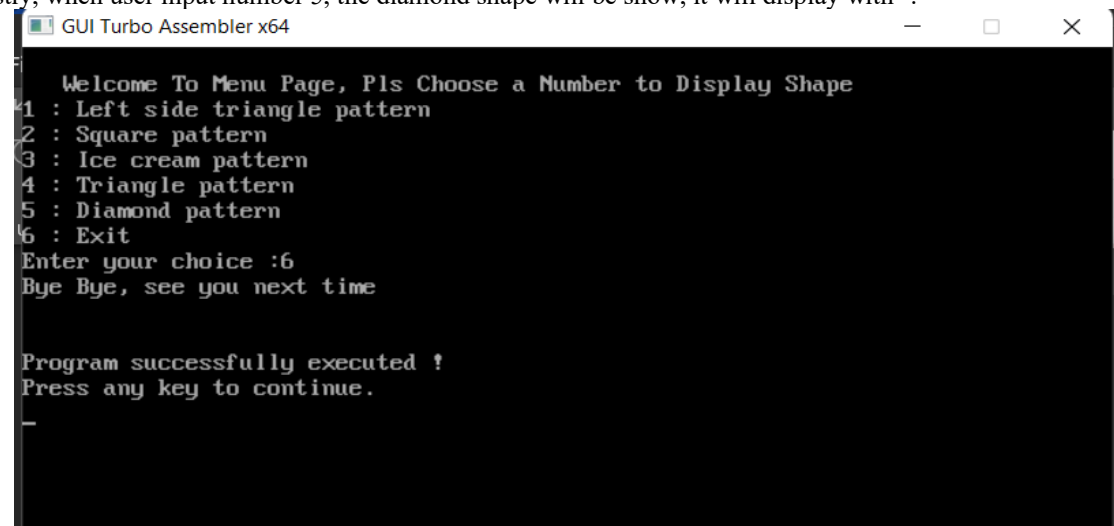
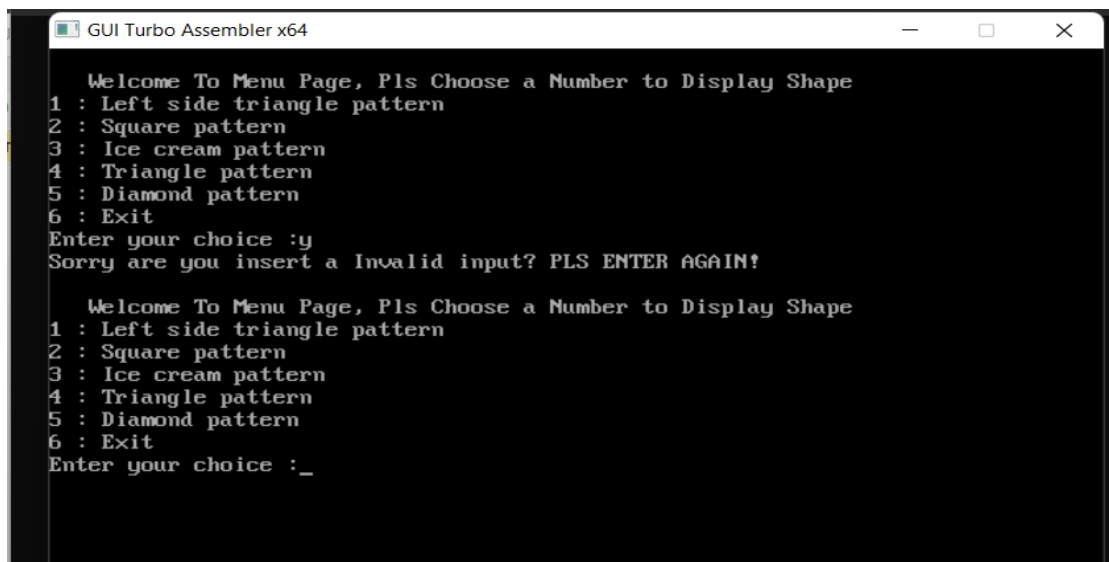


Figure 11 Display Bye message

When user input number 6, the system will direct exit, and the system will show the Bye message.



```
GUI Turbo Assembler x64

Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :y
Sorry are you insert a Invalid input? PLS ENTER AGAIN!

Welcome To Menu Page, Pls Choose a Number to Display Shape
1 : Left side triangle pattern
2 : Square pattern
3 : Ice cream pattern
4 : Triangle pattern
5 : Diamond pattern
6 : Exit
Enter your choice :_
```

Figure 12 error message when insert invalid input

When user input invalid input such as empty, alphabet, greater than 7, lower than 1.

6.0 Limitation

Assembly language has various drawbacks, one of which is that it takes far longer to create than high-level languages. Because assembly language is machine-understandable code, it is more challenging to interpret than other high-level programming. Maintainability is yet another assembly language drawback. It is because assembly code is hard to adapt and unstructured, making it challenging for others to comprehend. Reliability and security come last since it is simple to make a coding error when programming. For instance, if you are having trouble verifying the system's push and pop instruction count, nothing will appear. There are several ways that assembly code might be written incorrectly. Users must take a methodical approach to testing and validating since there are several opportunities for error in assembly code, which might have an impact on the project's dependability and security. (YOVAK, 2022)

There are certain restrictions when designing a decent form or a sophisticated shape for a program since it is difficult. As an illustration, when I initially get an idea for a form, I might place text in the center of a square that already has some color, but after I finish, all the square has is the symbol #, with no text or color within. The next step is to test the loop after creating a shape to check how much the symbol's increment will contribute. This is really difficult to check on the syntax to see why the shape is doesn't display out, so need use much of time to compile the program and run it, then only can figure out the code error.

7.0 Conclusion

As a result of its ability to translate between machine language and human language, assembly language is crucial for all programmers. Mnemonic and operands are still present since the machine can only understand binary code at this time. After doing the task, I gained knowledge and a greater comprehension of how to use the capabilities of assembly language. As an illustration, I created the main menu page and a shape in the assignment using the information. Even if most developers in today's world are more accustomed to high level programming and don't even know how to study assembly code, they risk losing part of their expertise if they do. Not only that, but I also gained knowledge of picture stenography and reverse engineering tools, both of which will be useful when I must deal with reverse engineering problems during a CTF challenge. Finally, mastering assembly language is crucial if you work in forensics or cyber security. It is because, the assembly language will assist the developer to understand the malware code and utilize it to avoid or solve the malicious onslaught.

8.0 Self Reflection

In my opinion, in this assignment I learned some new knowledge that will involve me become much better in a cyber security field. For example, I learned the how the assembly language work, and how to use it for coding. Not just that, this assignment I had learn the usage of IDA pro tools and some reverse engineering disassembler tool for identify malware. This will be very helpful, since am a cyber security student I had participate on the CTF which is capture the flag, by using the assembly language knowledge and the disassembler knowledge, maybe I can be easier to solve my challenge. Since when starting of the assembly language, I actually no idea how to read the assembly code, and don't even how to print the shape. But after getting some explanation of lecturer I become a bit familiar on the assembly language, then I had success to create 5 shapes for this assignment. Lastly, this assembly language let me know that assembly language is still usable even through is cyber security or forensic for prevent the cyber-attack.

9.0 References

- Aticleworld. (n.d.). *C vs Assembly language*. Retrieved from Aticleworld: <https://aticleworld.com/difference-between-c-and-assembly-language/>
- Ayush Kejariwal, S. B. (2019). Information Security Using Multimedia. *Smart Innovation, Systems and Technologies 153*, 643.
- Azu, R. (2020, July 31). *How to use Radare2 for reverse engineering*. Retrieved from INFOSEC: <https://resources.infosecinstitute.com/topic/how-to-use-radare2-for-reverse-engineering/>
- Baldawa, S. (2020, September 23). *Radare2 for reverse engineering-Part1*. Retrieved from ITNEXT: <https://itnext.io/radare2-for-reverse-engineering-part1-eedf0a47b5cc>
- BINARYNINJA. (2016). *Stable Branch Changelog*. Retrieved from BINARYNINJA: <https://binary.ninja/changelog/>
- byjus. (n.d.). *byjus*. Retrieved from High-Level Vs. Low-Level Languages: Find What is the Difference Between High-Level and Low-Level Languages? : <https://byjus.com/gate/difference-between-high-level-and-low-level-languages/>
- e-spincorp. (n.d.). *IDA Pro Interactive DisAssembler for Software Analysis*. Retrieved from e-spincorp: <https://www.e-spincorp.com/ida-pro-interactive-disassembler-for-software-analysis/>
- Fox, N. (2022, March 21). *How to Use Ghidra to Reverse Engineer Malware*. Retrieved from VARONIS: <https://www.varonis.com/blog/how-to-use-ghidra>
- GeeksforGeeks . (2021, August 20). *Difference between assembly language and high level language*. Retrieved from GeeksforGeeks : <https://www.geeksforgeeks.org/difference-between-assembly-language-and-high-level-language/>
- Granville, V. (2017, January). *Interesting Data Science Application: Steganography*. Retrieved from datasciencecentral: <https://www.datasciencecentral.com/interesting-data-science-application-steganography/>
- Lee, H. (2020). *New Approach on Steganalysis: Reverse-Engineering*. Retrieved from hkvisa: <https://sci-hub.hkvisa.net/10.1145/3384544.3384571>
- MenuetOS. (n.d.). *MenuetOS*. Retrieved from MenuetOS: <http://menuetos.net/>
- MKS075. (2020, may 13). *Difference between High Level and Low level languages*. Retrieved from geeksforgeeks: <https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/>

- Noerenberg, E. (2018). *Malware Analysis and Automation using Binary Ninja*. Retrieved from Irongeek: <https://www.irongeek.com/i.php?page=videos/bsidescharm2018/track-2-07-malware-analysis-and-automation-using-binary-ninja-erika-noerenberg>
- packetlabs. (2022, March 16). *What are Indicators of Compromise (IoCs)?* Retrieved from packetlabs: <https://www.packetlabs.net/posts/indicators-of-compromise-ioc/>
- pcmag. (n.d.). *QEMU*. Retrieved from pcmag: <https://www.pcmag.com/encyclopedia/term/qemu>
- Pedamkar, P. (n.d.). *What is Assembly Language*. Retrieved from educba: <https://www.educba.com/what-is-assembly-language/>
- Rapid7 . (n.d.). *Malware Attacks: Definition and Best Practices*. Retrieved from Rapid7 : <https://www.rapid7.com/fundamentals/malware-attacks/>
- researchgate. (2015, March). Retrieved from Virtualization for Cost-Effective Teaching of Assembly Language Programming: https://www.researchgate.net/publication/275029027_Virtualization_for_Cost-Effective_Teaching_of_Assembly_Language_Programming
- Shaid, S. Z. (2014). *INTRODUCTION TO MALWARE REVERSE ENGINEERING*. Retrieved from people.utm: <https://people.utm.my/syed/files/2014/06/bookchapter-intro-to-malware-reverse-engineering.pdf>
- YOVAK. (2022, january 19). *Advantages and Disadvantages of Assembly Language*. Retrieved from YOVAK: <https://yovak.com/advantages-and-disadvantages-of-assembly-language/>