**Logo, company name

Description automatically generated**

**INDIVIDUAL ASSIGNMENT.**

**CT07332 CSLLT**

**Computer System Low Level Techniques.**

**INVENTORY SYSTEM**

**HAND OUT DATE: 3RD WEEK.**

**HAND IN DATE: 13TH WEEK.**

**Shape**

INTAKE CODE: APD2F2309CS(CYB).

MODULE CODE: CT07332 CSLLT.

STUDENT NAME: ABDULRAHMAN GAMIL

MOHAMMED AHMED.

TPNUMBER: TP071012.

Contents

[1.0 Introduction to Assembly Language 3](#_Toc168550963)

[2.0 Research and Analysis 4](#_Toc168550964)

[2.1 Application of the Language in Forensic and Cybersecurity 4](#_Toc168550965)

[2.2 Analysis of Malware 5](#_Toc168550966)

[2.3 Reverse Engineering (RE) 5](#_Toc168550967)

[‎**2.4 Advantages‎** 7](#_Toc168550968)

[2.5 Challenges 7](#_Toc168550969)

[2.6 Modern Relevance 7](#_Toc168550970)

[2.7 Future Directions 7](#_Toc168550971)

[2.8 Key Features to Highlight In the Program 8](#_Toc168550972)

[3.0 System Design 9](#_Toc168550973)

[4.0 System Screenshots 12](#_Toc168550974)

[5.0 Source Code 23](#_Toc168550975)

[6.0 Conclusion 37](#_Toc168550976)

[7.0 References 38](#_Toc168550977)

# 1.0 Introduction to Assembly Language

As the most basic programming language, Assembly Language plays a central role in ‎the development of every computer system. It is a direct abstraction of machine language, making it ‎closer to the hardware than any other programming language. Assembly languages are used by ‎developers to code the operations of a computer's Central Processing Unit (CPU). ‎Luckily, programmers do not have to work directly with the binary form of machine ‎language. This complexity is replaced with a more readable language by converting mnemonics ‎into machine language. Opcodes are defined in a more memorable way (Thakur, 2023). For ‎example, the opcode 'mov' is a shorthand for 'move' and is used to transfer bits within ‎registers. This human readable nature of assembly code is a reason why developers love ‎it. The language is hardware-specific; thus, any program developed using it is more efficient ‎and faster than one developed in a high-level language. Regardless of many high level ‎programming languages being developed, Assembly Language is the root translator for all ‎programming languages. Its flexibility and full control over the hardware give the best human ‎interface to program a computer (Kanade, 2023).‎

With this understanding of Assembly Language, the paper shall delve into its application in ‎the cybersecurity domain. Digital forensics, malware analysis, and reverse engineering ‎procedures have a dependence on a deep understanding of the Assembly language. Therefore, ‎this paper will dissect an executable program by analyzing the assembly code with examples for ‎clarity on its understanding. In addition, there is documentation of an Assembly program designed to manage inventory, a ‎flowchart explaining the program, a user guide, and key code snippets for further reference.‎

# 2.0 Research and Analysis

Assembly language is one of the very basic tools for computer programming that allows a ‎very close to the mental approach to software development. This study and analysis focus on the key ‎aspects, applications, and challenges of assembly language.‎

2.1 Application of the Language in Forensic and Cybersecurity

Assembly language plays a vital role in cybersecurity and forensics because it gives experts the ability to carefully deconstruct, inspect, and understand dangerous code and software that may have complex inner operations. This low-level language is vital for malware investigation, especially when using debuggers to unearth code flaws and reverse-engineer. Additionally, assembly language improves dynamic analysis by keeping an eye on memory and system states in real time. In that way, forensic investigations get to find any malicious activities and security flaws. All these, in an ever-changing world of cybersecurity and digital forensics, go a long way in ensuring a key role in performing vulnerability assessment and root cause analysis toward coming up with very strong security measures that will help guarantee the integrity of computer systems and networks

2.2 Analysis of Malware

Assembling malicious code is one of the key objectives of assembly language for the purpose of malware ‏investigation. Cybersecurity professionals need to reverse engineer malicious software using a disassembler like HexRays IDA to understand its behavior and discover any concealed malicious activities from the software ‏(Shaid, 2014). The low-level, human-readable form that the assembly language provides makes the interpretation of the malware's actions easier.

Moreover, dynamic malware analysis—watching the behavior of the infection as it operates— also thrives on assembly language. Assembly language is what debuggers use to display the state of the system at that time, including all registers and memory, making it very informative on how the virus is talking with the host system (Cybersecurity Malaysia, 2023). In addition, the virus code is thoroughly inspected by cybersecurity professionals with the help of assembly language. The possible methods, reasoning, or weaknesses that the malware might have can thus be obtained, which is crucially important information for the creation of patches or countermeasures to reduce the risk (IBM, 2023).

2.3 Reverse Engineering (RE)

In the context of assembly language, reverse engineering is the act of taking an ‎executable programme, or binary file, apart to see what makes it tick and perhaps change its ‎behavior. ‎Lowlevel programming languages such as assembly language are quite close to the machine ‎code that a computer's central processing unit (CPU) executes. Reverse engineering often ‎involves disassembling a binary file to recreate the assembly code that produced it, when the ‎original source code is not available but needs to be adjusted or debugged. (eForensics, 2022).‎

Binary analysis can be done with assembly language through the use of reverse engineering ‎techniques. This often includes the study of compiled binaries to deduce the way the code works. ‎Reverse engineers can study and comprehend the way the program works because assembly ‎language gives those binaries a human readable form (Vickie, 2020). Assembly language is utilized ‎in reverse engineering under digital forensics and cybersecurity to study and modify software, a ‎process that is, at times essential during software cracking. It enables the bypassing of licensing ‎problems or the discovery of bugs (GeekBits, 2022). Forensic analysis also utilizes assembly in the ‎reverse engineering of potentially malicious software. Based on Arasteh and Debbabi (2007), this ‎analysis helps in narrowing down the attack, identifying methods to steal data, and allowing the ‎attack to be traced to specific threat actors.‎

Assembly language was designed in the first years of the computer to overcome the problems ‎of programming in machine code—the binary language that a computer's CPU directly ‎understands. It emerged as a more readable and manageable alternative, with mnemonics to ‎represent machine instructions and symbolic names for memory addresses. The earliest ‎computers, for example, the IBM 704 and UNIVAC, used assembly languages developed for their ‎particular architecture.‎

1. System Programming: Assembly language is critical in the writing of operating systems, device drivers, and embedded systems. It provides perfect control over hardware that is important in handling low level activities and making sure that everything performs efficiently.
2. System Programming: Assembly language is vital in writing operating systems, ‎device drivers, and embedded systems. It provides perfect control over the hardware, ‎which is important in handling low level activities and ensuring that everything ‎performs efficiently.‎
3. ‎ Performance critical Software: Assembly language makes it possible for the writing of ‎high performance applications, such as video games and scientific simulations, due to its ‎speed and efficiency.‎
4. ‎ Hardware Interaction: Assembly language, very useful in dealing with hardware ‎components like CPU registers, memory, and I/O ports, is important for interfacing ‎hardware and low level debugging.‎
5. ‎ Educational Tool: The other major use of assembly language is in teaching computer ‎science, with computer architecture, operating systems, and the low level basics of ‎programming.‎

‎2.4 Advantages‎

• Performance: The assembly language programs are quite efficient at runtime, and memory usage is optimal because they can take advantage of particular hardware features and instruction sets.

• ‎Control: Allows for granular control over the hardware, so programmers can fine tune ‎operations and manage resources precisely.‎

• Predictability: Assembly language code can be deterministic and predictable—what is ‎crucial in real time systems where the importance is laid on time.‎

2.5 Challenges

• Complexity: Writing in assembly language is complex and time consuming compared to ‎‎high-level languages. Writing in this language requires a deep understanding of the hardware architecture and ‎‎the instruction set.‎

•t‎Portability: Assembly language is not portable. In other words, one must rewrite ‎programs for different CPU architectures, which limits using it for cross platform ‎development.‎

• ‎Maintainability: Assembly code is less readable, understandable, and maintainable. This ‎increases development time and is error-prone.‎

2.6 Modern Relevance

Despite the rise of high level programming languages, assembly language remains relevant in ‎particular areas. Its use in embedded systems, performance critical applications, and ‎hardware interfacing remains to be significant. Improvements in tools and assemblers have made the assembly language more accessible, enabling programming in ‎assembly ‎routines within ‎high level ‎language ‎programs to realize performance optimization. ‎

2.7 Future Directions

The future of assembly language lies in its integration with high-level languages to provide the best of both worlds: the control and efficiency of assembly with the ease and productivity of high-level languages. Research in automated assembly code generation and optimization keeps evolving, potentially making it more accessible and less complex to write and maintain assembly programs.

An implementation of an Inventory Management System using assembly language is one of the most vital practical applications in the use of low-level programming and its relevance. This goes on to show how efficient and accurate assembly language can be, especially in solving complex processes such as inventory tracking and sale processing, by using the hardware directly and increasing performance.‎

On this task, both the challenges and the benefits of working in assembly language were evident. Attention to detail, a deep understanding of hardware nuances, and the ability to effectively manage system resources played a pivotal role in the realization of project objectives. Even though the detail was quite complicated, the use of assembly language made it possible to design a system that was highly effective and used direct hardware manipulation to achieve better performance.‎

This implementation using assembly language serves as evidence that it is still an applicable tool for efficiency-demanding and closely hardware-interacting applications. Although mainly the high-level languages are used because of their ease, the experience in using the assembly language very much reinforces the understanding of computer architecture and the principles of low-level programming.

2.8 Key Features to Highlight In the Program

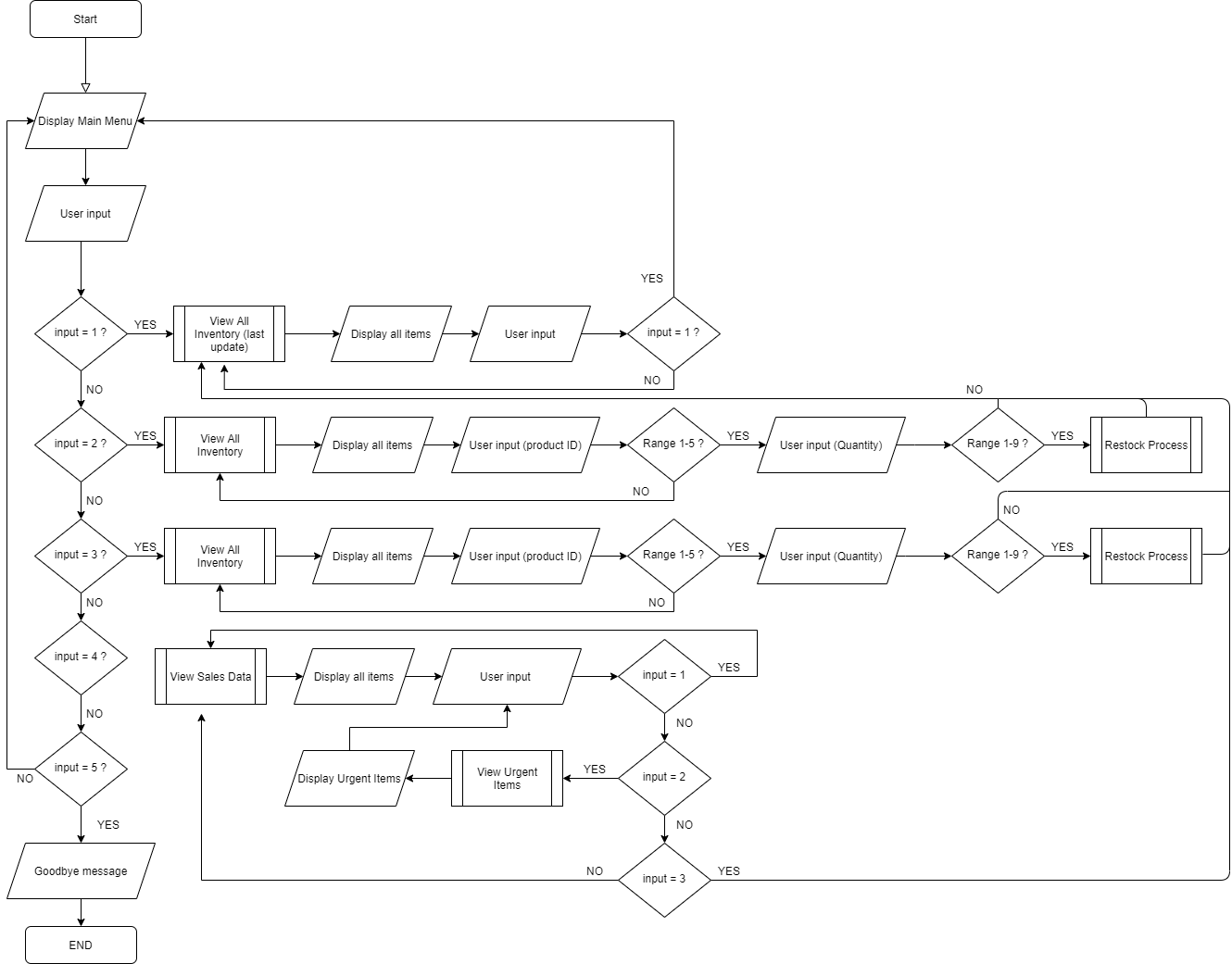
• UserFriendly Interface: It has a simple menudriven interface that makes it easy for ‎users to use the system, even when they are not conversant with assembly language.‎

• Modular Design: Every option in the menu corresponds to a specific function, ‎modularizes the tasks and makes the system easy to maintain and update.‎

• RealTime Updates: As users add stock or process sales, the system updates the ‎inventory in real time, which guarantees that the data is up-to-date.‎

•tCritical Alerts: The critical alert choice must have the option for stock and sales data, ‎alerting for low stock items, and users must take timely actions to restock.‎

# 3.0 System Design



1

**‎1. Begin:‎**

‎ The process begins in the "Start" node.‎

**‎2. Present Main Menu:‎**

‎ The system presents the user with the main menu options.‎

**‎3. Accept User Input:‎**

‎ The user is asked to enter a choice from the main menu.

‎**4. Menu Items:‎**

‎ The user enters a number between 1 and 5 to have the menu option selected‎

‎ Option 1: List All Inventory

‎ If the user enters 1, the system lists all items in the inventory along with the quantity for each item‎

‎ The user will be asked to enter a product id to re-stock the product‎

‎ If the product ID is in the valid range, the user enters the quantity to be restocked‎

‎ If the quantity is in the valid range, the re-stocking process will be performed‎

‎ If either product id or quantity is out of range, the user is asked to re-enter the data ‎

‎ Option 2: Display All Inventory

‎ If the user enters 2, the system will display all the inventory items‎

‎ The user will be asked to enter a product id to view the product‎

‎ If the product ID is in the valid range, the system displays the information related to the specified product‎

‎ If the product ID is out of the range, the user is asked to re-enter the data ‎

‎ Option 3: Display All Inventory (different path)‎

‎ If the user enters 3, the system will again display all the inventory items‎

‎ The user will be asked to enter a product id to view the product‎

‎ If the product ID is in the valid range, the system displays the information related to the specified product ‎

‎ If the product ID is out of range, the user is prompted to input the data again.‎

‎ Option 4: View Sales Data

‎ If the user enters 4, the system displays all the data in the sales.‎

‎ The user is prompted to enter more options.‎

‎ If the user enters 1, the system displays urgent item.‎

‎ If the user enters 2, the system views urgent items.‎

‎ If the user enters 3, the system goes back to the main menu.‎

**‎ Option 5: Exit**

‎ If the user enters 5, the system displays a goodbye message and terminates the process.‎

**5. Farewell Message:**

‎‎ If the user selects exit, the system displays a goodbye message.

‎**6. End:‎**

‎ The process stops at the "End node.‎

The flowchart outlines an inventory management system for users to check all inventory, restock ‎items, view details about a specific product, view sales data, and display urgent items. Before the user proceeds to restock ‎an item or view details, the system should guarantee that the product ID and quantity are valid. An option to gracefully exit the system is provided.

# 4.0 System Screenshots

A screenshot of a computer

Description automatically generated

2

This is the main menu screen of your inventory management system, where users are presented ‎with several options to interact with the inventory.‎

**‎1. Welcome and User Information:‎**

‎ The top section of the screen shows a welcome message along with your name "Abdulrahman ‎Gamil" and an identifier "TP071012". This is more personalized and helps in ‎tracking usage or access in a multi-user environment.‎

**‎ 2. System Structure Menu: ‎**

‎Under the welcome message, there is the 'System Structure' header followed by the structured menu of the system. This menu is the one that leads the user to its functionalities of the inventory management system.

**3. Menu Options:**

The menu lists five options:‎

‎ 1. View Products: This option allows users to view the current list of products in the ‎inventory.‎

‎ 2. Add Stock: This option lets users add new products or increase the stock of existing ‎products.‎

‎ 3. Process Sale: This option is used for recording sales transactions and reducing the ‎inventory accordingly.‎

‎ 4. Critical Stock & Sales Data: This option provides critical information about stock levels ‎and sales data, which is useful for inventory management and decision making.‎

‎ 5. Quit: This option allows users to exit the system safely.‎

**4. User Prompt:**

At the bottom, you have a prompt "Please make a selection:", which gives a ‎message that the system is waiting for the user to make a selection by entering the number corresponding to the selected option.‎

Your inventory management system offers a clear and structured menu, enabling the user to easily ‎find their way around it. Options provided are to ensure particular aspects of inventory ‎management are smoothly carried out: manage products, track sales, and always keep optimal stock ‎levels. ‎

A screenshot of a computer

Description automatically generated

3

**1. View Commodities:**

When the user chooses this option, the system will list all the products that are in the inventory at ‎the moment. The list may comprise data such as the product name, product ID, available amount, and ‎price. This way, the user can have an up-to-date overview of the inventory.‎

Inventory Information Screen

The system will present an overview of the detailed information about the current inventory ‎quantity, price details, and sales performance for each of the products. ‎The header of the inventory information screen will include clear information as ‎Inventory Details. The major ‎body of the screen will include a product ID, name, available stock quantity, product price, and sold ‎quantity in a tabular form. This table is color coded for easy identification of stock levels, where ‎colour red is used for identifying products with very low stock levels that urgently require restocking, and ‎green is used for identifying products with sufficient stock levels. For instance, 'Shelf' and 'Panel' are ‎products that are in red, which means their quantity in stock is not enough, while 'Brush', 'Box', and ‎'Jars' are in green. Below the table is a statement on the color-coding for easy reference. B. At ‎bottom, the user can enter an option '1' which will let the user go back to the main menu, and this ‎helps to navigate back to the main menu when the user has to conduct further activities.‎. This screen will serve to enable

A screenshot of a computer

Description automatically generated

4

**2. Add Stock:**

‎Choosing this option will provide users with a screen where they can input the necessary details ‎to add new items or update the stock of already added items. The system may require the product ‎ID, name of the product, and the quantity to be added, and the price per unit. Input validations ‎must be taken care of in the system, and successful addition of stock needs to be confirmed.‎

A screenshot of a computer

Description automatically generated

5

A screenshot of a computer

Description automatically generated

6

Upon pressing "2" from the main menu, the user will be directed to the inventory information screen, which contains information regarding products' IDs, names, stock levels, prices, and number of units sold. This screen is located under the label "inventory info." The stock levels are color-coded for easy understanding—if the stock is 3 or more, then the number is green; if the stock is less than 3, then the number is red. At the bottom of the screen, the program prompts the user to enter the ID of a product to add to its stock. In the image below, we have typed in ID ‎‏1‏‎ to add some more of the stock from the product "Panel." We then added 2 more units to the product. After updating, the inventory information shows the changes; the stock level for product ID 1 has been updated. Users will press "1" to return to the main menu after completing the "Add" function. This is the feature to ensure that after completing the "Add" function, users can easily navigate back to the main menu.A screenshot of a computer

Description automatically generated

7

**3. Process Sale:**

This provides an option where a user can record a sale transaction. The user must probably enter the product ID, quantity sold, and maybe customer information. The system then updates the inventory by reducing the stock of the sold items and maybe record the details of the transaction for receipt generation or tracking of the sales.A screenshot of a computer

Description automatically generated

8

A screenshot of a computer

Description automatically generated

9

After clicking on "3" on the main menu, you access the Process Sale function. The purpose ‎of this function is to reduce the ‎quantity of a product in stock, as specified by the user. ‎This screen is similar to the Add Stock screen.‎ In the example provided, we entered the ID of product ‎"Panel" (ID 1) in this process sale screen to ‎process a sale. The user then enters a quantity that is sold. In this example, we recorded the sale of ‎3 units. ‎The software will then reduce the stock level for that particular product (ID 1) by the quantity ‎sold. It then updates the inventory, reflecting the changes brought about as a result of the sale ‎‎.‎

‎

Clicking on the "option 1" at the base of the screen ‎leads you back to the main menu, from where one ‎can perform other tasks.

.‎A screenshot of a computer

Description automatically generated

10

**4. Critical Inventory & Sales Information:**

It creates a provision of some important data pertaining to the inventory, like items that are low ‎in stock, total sales, and possibly trends in sales data. This helps users find out what ‎needs restocking and analyze the performance of different products.‎

A screen shot of a computer

Description automatically generated

11

A screenshot of a computer

Description automatically generated

12

Once we selected option 4, "Critical Inventory & Sales Information," we were offered a good ‎presentation of the sales data. It presents to us the product ID, name, price in RM, ‎quantities/units being sold, and total profit. The stocks with less than 3 units remaining are, by ‎default, being hidden in order to make things clearer. The display of items like this can be ‎obtained by pressing 2, which will bring us to "Examine Critical Products," where all items ‎with quantities below 3 are presented. One can then go back to the main menu by selecting 3 and to the page displaying Sales Data by selecting 1.‎A screenshot of a computer

Description automatically generated

13

**5. Terminate Program:**

This option allows users to exit the system. The system should ensure all data is saved properly when the program is closed so that data is not lost.

A black and white logo

Description automatically generated

14

The last feature, "Terminate Program," is a feature that allows a user to gracefully exit ‎the program from the main menu. If the user chooses option 5, the program is terminated ‎and a message is displayed: "Thanks for utilizing the inventory management system!"‎

# 5.0 Source Code

A screen shot of a computer program

Description automatically generated

15

The above screenshot depicts data for five commodities in assembly language, which describes the attributes of ‎each product in a tabular manner. Each product is comprised of an ID, Name, Stock, Price, Priority Level, ‎Products Sold, and Currency. The `dw` (define word) is used for numerical values, while the `db` (define byte) is applied for text. Such a structure allows for logical storage and straightforward access to product information from within an assembly language program.

A computer screen shot of a program

Description automatically generated

16

The `showcaseCommodity` macro is designed to display various details of a product by calling ‎a series of functions and manipulating registers to process and show the data. It begins by ‎printing a new line to format the output, then initializes the base pointer to zero and loads the ‎offset address of the product data into the SI register. The macro then loads the product ID into ‎the AX register, converts it to an integer, and prints a space. Next, it calculates the full address ‎of the product name and calls a function to display it. For the stock, the macro loads the stock ‎value, checks if it's low with a function call, converts the stock to an integer, and prints a space. ‎Following this, the macro retrieves the product price, converts it to an integer, and prints a ‎space. Lastly, it loads the product sales, converts them to an integer, and displays the final ‎formatted output. This process effectively retrieves and displays a product's ID, name, stock, ‎price, and sales by leveraging function calls for conversion and display, ensuring modularity ‎and readability.

A screen shot of a computer program

Description automatically generated

17

The above macro is a restocking request from the user for a specific product. It reads the amount to restock, converts it from ASCII to a numerical value, updates the inventory, and displays a success message. Finally, it returns to the inventory menu.

A screenshot of a computer program

Description automatically generated

18

This macro processes Commodity data in great detail, converting string representations of names, prices, and sales to integers for the purpose of computation. It multiplies the price and sales to make an accurate estimation of the total revenue brought in by each respective product, giving valuable insight into the financial performance of this inventory.

A screenshot of a computer

Description automatically generated

19

This macro efficiently handles the selling process by asking for the number of items to be ‎sold, converting it into a number data type, and then subtracting the same from the available ‎stock, ensuring if the stock has depleted. Once it has updated the stock, it moves on to ‎showing a success message for a complete sale.‎

A computer screen shot of a program

Description automatically generated

20

The "TransactionSuccessful" macro clears ‎the window, updates the amount of sales, ‎prints a message of success, and prompts the user for further actions, such as displaying products and taking input. In addition, it will handle going to display the failure message if the stock is insufficient.

A computer screen shot of a program code

Description automatically generated

21

The "SaleReset\_" macro processes the situation of a failed sale: it clears the window, restores ‎the original stock value, presents a message of failure to the user, and provides options for further ‎actions, such as presentation of products, and processing additional user input. Finally, it returns control to the calling function.‎

A screenshot of a computer program

Description automatically generated

22

This macro clears the window, displays all products, then checks each product's stock. If it's lower than 3, it displays the product. Finally, it calls a function to complete the display of filtered products and returns.

A screenshot of a computer program

Description automatically generated23

This primary process begins by initializing the data segment and printing the main menu. ‎It waits for the user to make a selection then reads a character and sends it to the appropriate ‎handler: inventory, restock, sale, category list, or exit. If none of these match, it will start over ‎and call the primary process again.‎

A screenshot of a computer program

Description automatically generated

24

The InventoryMenu\_ shows the products and requests user input. Restock\_Menu\_ requests restocking, Sales\_Menu\_ requests sales, \_CategScreen\_ categories of products, and Terminate ‎exits the program. Each of these windows, before producing their respective outputs, clears the screen. TerminateSys\_ is an orderly exit of the system and produces a line before exiting.

A screenshot of a computer program

Description automatically generated

25

The DisplayLine\_ function outputs a separator line. The ClearTheWindow function clears the ‎window, and UserinputDecision handles both the input of the menu and the displaying of the options ‎and reading the user's choice. It directs to the relevant menu or exits based on user input.

A screenshot of a computer program

Description automatically generated

26

The "CategMenu\_" function gives the user category options. It reads input and directs to ‎corresponding actions: showing the full inventory, showing low stock products, or returning to the main menu‎. A screenshot of a computer program

Description automatically generated

27

The SmallStockCheck\_ function is used to compare a value (in AX) to 2 and if its less than or ‎equal, show it in red, and in green if not. The ConvertStringToInteger function converts a ‎string to an integer using BX as the base.‎

A screenshot of a computer program

Description automatically generated

28

This assembly code extract above presents a menu for the user to select between three ‎choices. Depending on the input taken from the user ('1', '2', or '3'), it jumps to the respective ‎subroutines: `FullInventoryshowcase\_`, `Commodities\_tockL\_`, or `skipInvalidInput3\_`). Invalid ‎input redirects to the `Commodities\_tockL\_` routine, which keeps monitoring low stock.‎



29

"ENDP MAIN" marks the end of the main procedure, and "END MAIN" indicates the end of the program. Everything between "MAIN PROC" and "ENDP MAIN" constitutes the main program logic.

# 6.0 Conclusion

Assembly Language plays a critical role in computer programming and cybersecurity. It acts as a link between high-level programming and low-level machine instructions, and hence it ensures that software works in coordination with hardware at the lowest levels possible, and in a manner that assures the best performance and efficiency. Very close to machine language, it is indispensable for efficient hardware interaction.

In real time systems, Assembly Language is particularly valuable for ensuring precise and ‎immediate responses. It is extensively utilized in critical industries, as discussed in this report. ‎The report also underscores the significant role of Assembly Language in cybersecurity and ‎digital forensics, highlighting its importance in malware analysis and reverse engineering. ‎

Cybersecurity experts rely on Assembly Language to disassemble malicious code, reverse ‎engineer binaries, and detect concealed malicious activities, enabling the development of ‎effective countermeasures and enhanced security protocols. Overall, Assembly Language ‎remains a fundamental tool in programming, cybersecurity, and digital forensics, contributing ‎to efficient software development, thorough security analysis, and system optimization. Its ‎low level, human readable format makes it indispensable for professionals in these fields.‎

# 7.0 References

* Arasteh, A. R., & Debbabi, M. (2007). Forensic memory analysis: From stack and code to execution history. Digital Investigation, 4, 114–125. <https://doi.org/10.1016/j.diin.2007.06.010>
* Assembly Language working, features, and Advantages | Spiceworks Spiceworks. (n.d.). Spiceworks. https://www.spiceworks.com/tech/techgeneral/articles/whatisassembly language/
* Fernando, J. (2023, July 27). Assembly language. Investopedia. https://www.investopedia.com/terms/a/assemblylanguage.asp#:~:text=Assembly%20lanuage%20is%20used%20to,such%20as%20high%20frequency%20trading.
* GeeksforGeeks. (2023, April 4). Interaction of a Program with Hardware. [https://www.geeksforgeeks.org/interactionofaprogramwithhardware/](https://www.geeksforgeeks.org/interaction-of-a-program-with-hardware/)
* Li, V. (2021, December 14). Intro to Reverse Engineering, Part 2 The Startup Medium. Medium. [https://medium.com/swlh/introtoreverseengineeringpart24087a70104e9](https://medium.com/swlh/intro-to-reverse-engineering-part-2-4087a70104e9)
* Machine Language vs. Assembly Language | Spiceworks Spiceworks. (2023, April 18). Spiceworks. https://www.spiceworks.com/tech/techgeneral/articles/machinevsassembly language
* Papa, K. (2023, October 9). Assembly RealTime Systems And Their Functions. MarketSplash. https://marketsplash.com/tutorials/assembly/assemblyrealtimesystems/
* Pedamkar, P. (2023, August 18). What is Assembly Language? EDUCBA. [https://www.educba.com/whatisassemblylanguage/](https://www.educba.com/what-is-assembly-language/)
* Salem, C. (2023, February 15). Introduction to Reverse Engineering What is Assembly Code? GeekBits. [https://www.geekbits.io/introductiontoreverseengineering/](https://www.geekbits.io/introduction-to-reverse-engineering/)
* Taylor, C. (2023, February 9). Assembly language. CyberHoot. [https://cyberhoot.com/cybrary/assemblylanguage/](https://cyberhoot.com/cybrary/assembly-language/)
* Thakur, S. (2023, August 18). Difference Between Machine Language & Assembly Language. unstop. https://unstop.com/blog/differencebetweenmachinelanguageandassembly language