Individual Assignment

CT073-3-2-CSLLT

COMPUTER SYSTEM LOW LEVEL TECHNIQUES

APD2F2409CS(CYB)

HAND OUT DATE: 19 March 2025

HAND IN DATE: 20 May 2025

WEIGHTAGE: 60%

LECTURE NAME: TS. UMAPATHY EAGANATHAN

NAME: PHURIPAT TANAPRADITKUL

TP NUMBER: TP071572

Contents

1.0 Introduction to Assembly Language

Assembly language is a low-level programming language that provides direct control over a computer's hardware. It is closely tied to the architecture of the machine and uses mnemonics to represent fundamental operations such as moving data, arithmetic, and logic. Unlike high-level languages, assembly offers precise control over CPU registers, memory addressing, and I/O operations, making it ideal for performance-critical and system-level programming.

In the context of this project, the Electronics Store Inventory System is developed using 16-bit x86 Assembly Language in the TASM (Turbo Assembler) environment. The purpose of choosing assembly is to demonstrate low-level programming proficiency while managing a complete inventory workflow — including viewing, restocking, selling items, and generating sales reports. Through this system, the power and limitations of assembly language become evident, as it requires meticulous handling of data and efficient logic structuring.

This project not only showcases the practical application of assembly language in managing real-world tasks but also deepens understanding of how hardware and software interact at a fundamental level.

2.0 Research and Analysis

Assembly language plays a foundational role in cybersecurity and digital forensics due to its close interaction with hardware and low-level system functions. Unlike high-level programming languages, Assembly provides direct control over system resources, memory addresses, and processor instructions, which are essential when analyzing malware, exploiting vulnerabilities, or performing reverse engineering (Sikorski & Honig, 2012).

In the field of cybersecurity, threat actors often write shellcode or critical exploit payloads in Assembly to bypass detection mechanisms. Security professionals must therefore understand Assembly to disassemble and analyze such threats using tools like IDA Pro or Ghidra (Eilam, 2011). This knowledge enables analysts to trace exact system behavior during an attack, such as how a buffer overflow modifies the instruction pointer or which registers are manipulated during execution.

In digital forensics, Assembly is critical for investigating incidents at the binary level. For example, memory dumps and executable files are often analyzed using Assembly to detect injected code or hidden payloads. Forensic investigators rely on disassembly techniques to reconstruct program logic or confirm tampering, especially when high-level source code is unavailable (Casey, 2011).

Moreover, capture-the-flag (CTF) competitions and penetration testing labs emphasize the use of Assembly for cracking software, bypassing authentication, and understanding system-level bugs. Mastery of Assembly enhances one’s ability to write more efficient exploit code and analyze compiled malware binaries effectively (Ligh et al., 2014).

The use of Assembly in the Electronics Store System project demonstrates these concepts in a simplified and controlled environment. Implementing logical operations, loops, conditions, and system interactions at the register level offers hands-on insight into how programs operate beneath the abstraction layer of modern languages.

Overall, proficiency in Assembly is a crucial asset for cybersecurity and forensic practitioners. It equips analysts with the skills to interpret binary behavior, respond to threats at the machine level, and develop secure, efficient low-level programs.

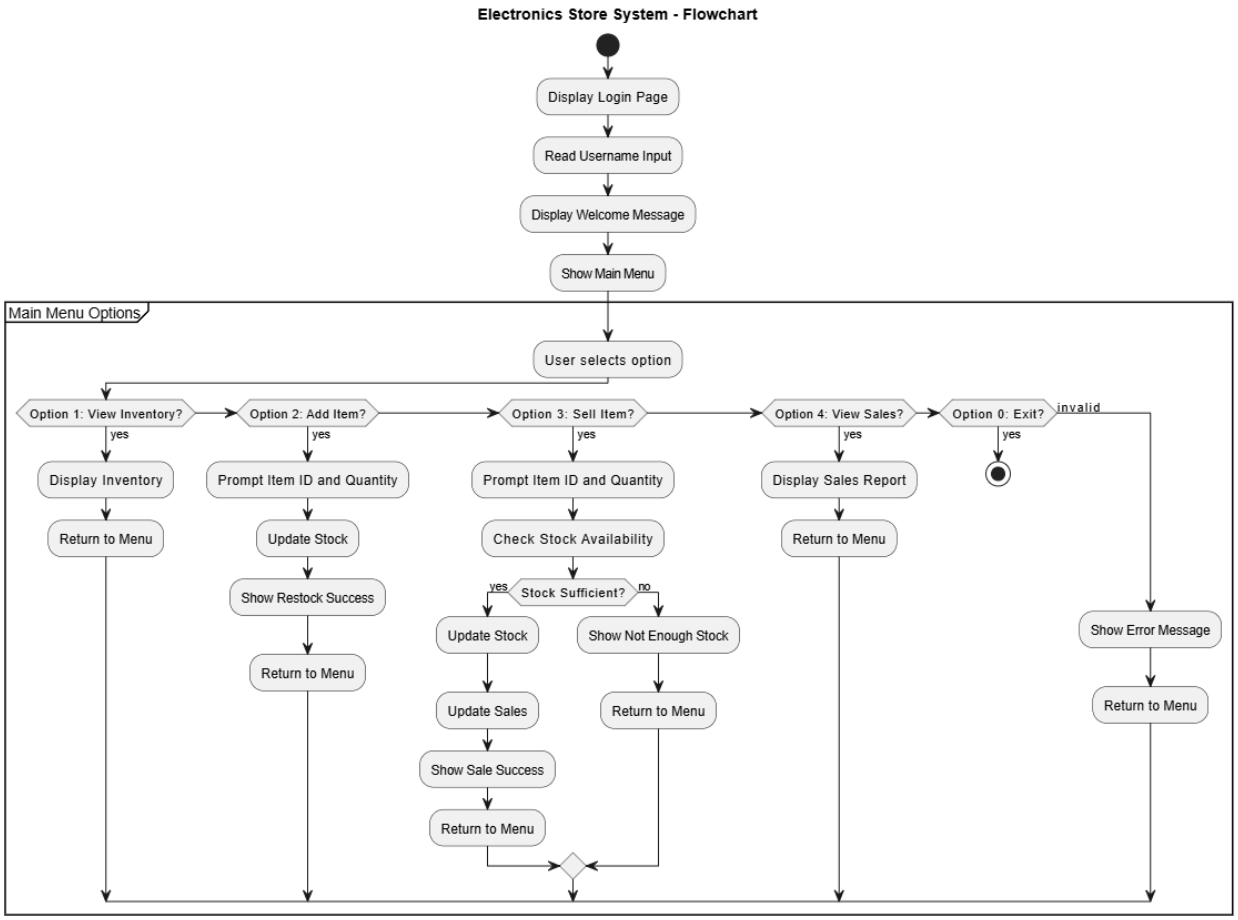
3.0 System Design with Flowchart

The design of the Electronics Store System follows a modular and user-centric approach. The system provides an interactive menu-driven interface that allows users to manage inventory, add new items, sell products, and view sales reports. Each function is separated logically, ensuring clarity in execution flow and maintainability.

The core components of the system are

- Login and Greeting: Verifies user identity and displays a customized welcome message.
- Main Menu Loop: Provides navigation to core functionalities.
- Inventory Display: Shows current stock with visual low-stock indicators.
- Restock Function: Adds quantity to existing items based on user input.
- Sell Function: Validates stock, updates quantity, and records total sales.
- Sales Report: Displays detailed breakdown of items sold and calculates total revenue.

System Design with UML Flowchart

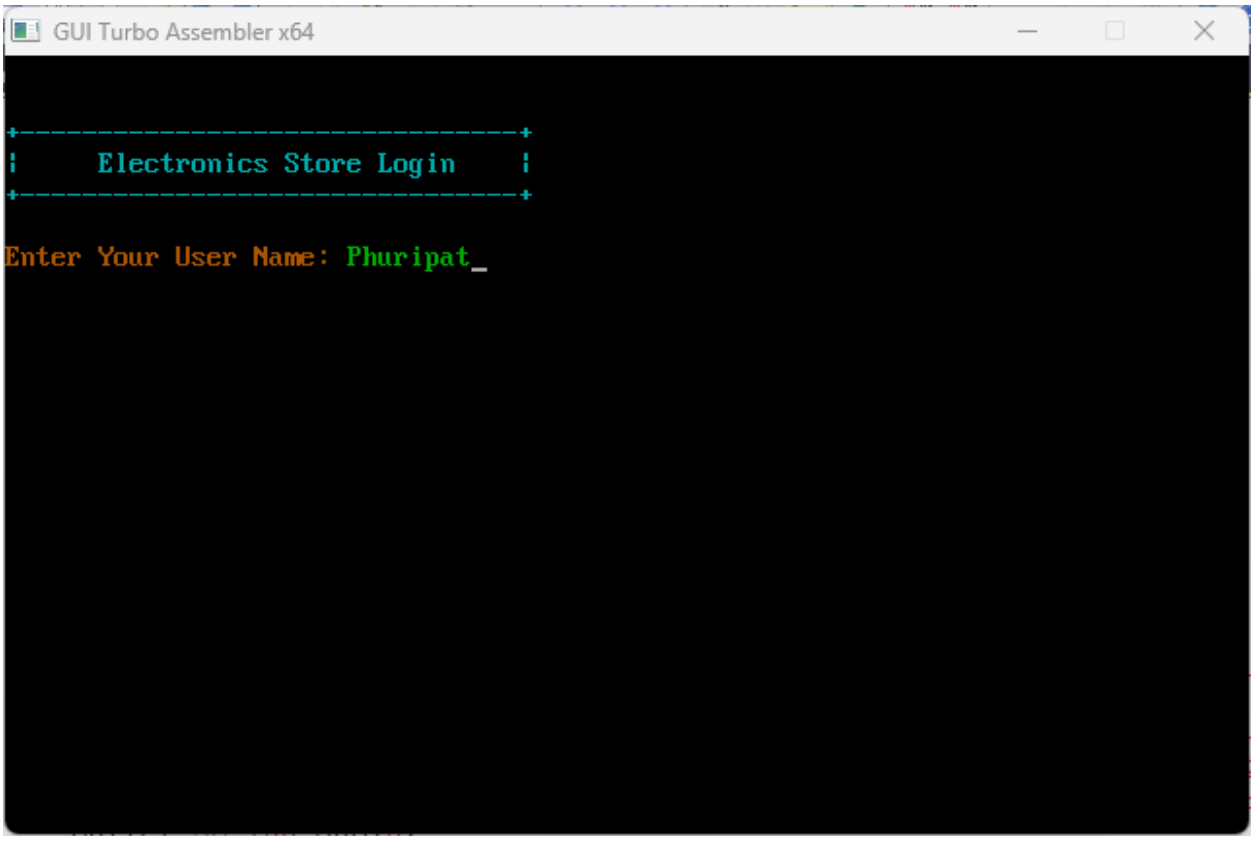


4.0: System Screenshot – Detailed List

This section presents visual evidence of the Electronics Store Inventory System in operation. Each screenshot illustrates a specific feature or interaction, with emphasis on the colored text output and intuitive layout designed to enhance usability and system transparency.

4.1 Login System

Function: Prompts the user for their username to initiate access.



- Features a visually styled header in cyan with the text "Electronics Store Login".
- Prompts the user to enter their username using yellow-colored text followed by a green input area.
- Provides a professional entry point into the system.

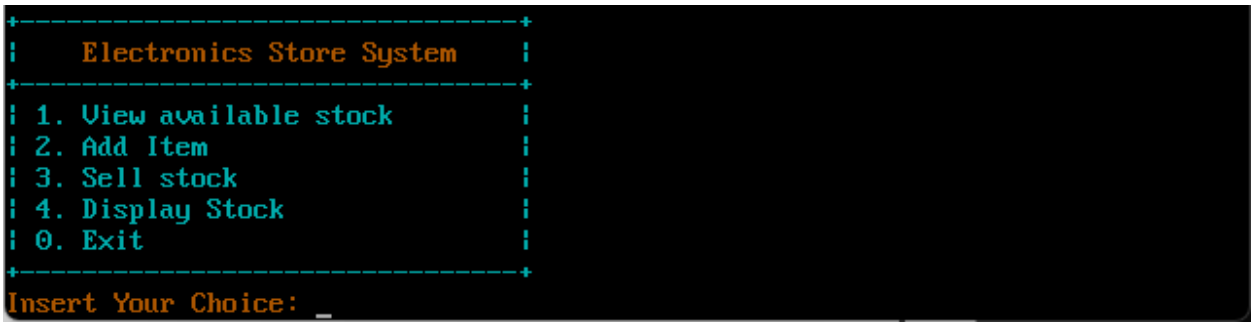
4.2 ASCII Art Robot Welcome

Function: Shows an animated robot mascot alongside the store's branding before main interactions.

- Displays detailed ASCII art of two symmetrical robot mascots.
- The robots flank a bling welcome banner that reads: "ELECTRONICS MEGA MART – Tech for Every Lifestyle!"
- This screen combines cyan, magenta, and blinking yellow for vibrant appeal.
- Sets the branding tone before entering the main interface.

4.3 Main Menu Interface

Function: Displays the navigational menu of system options.

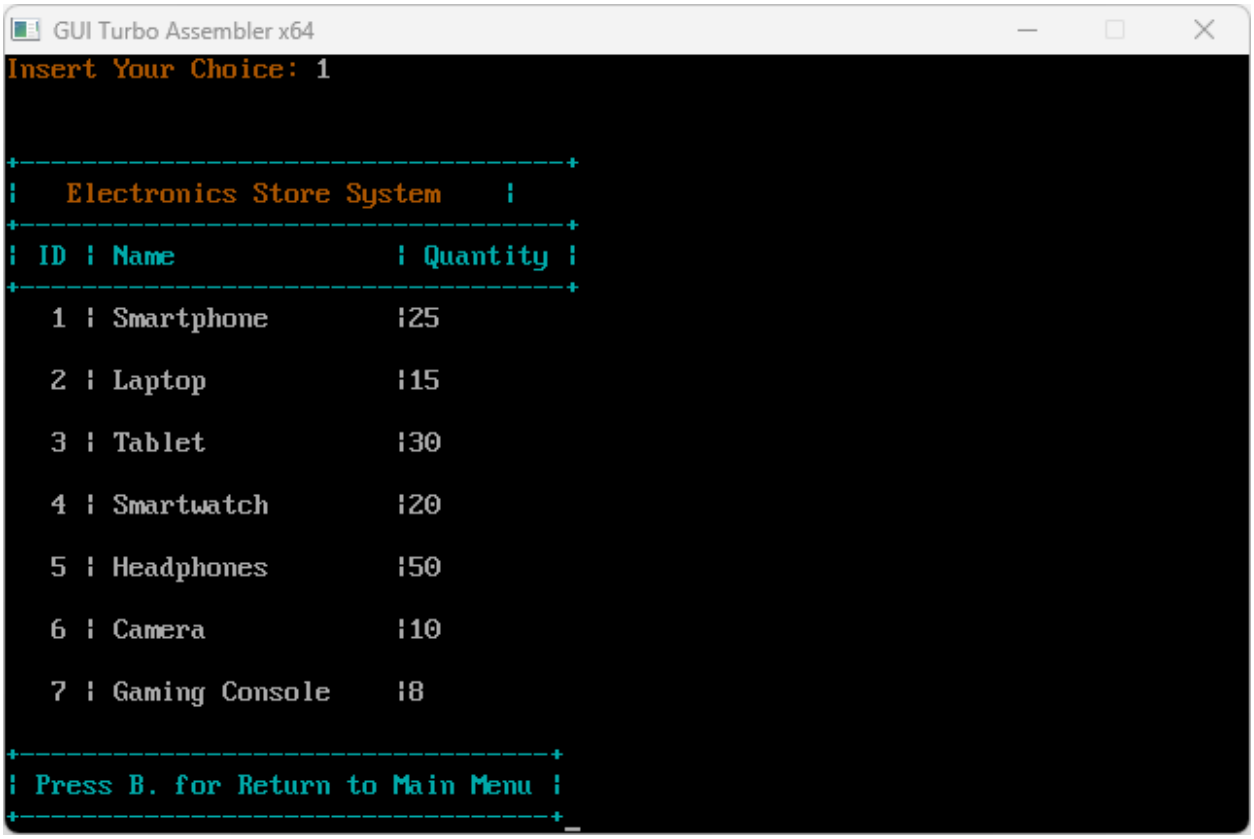


- Provides a clearly bordered and centered menu.
- Includes the five functional options: View available stock, Add Item, Sell stock, Display Sales Report, Exit.

- Yellow and green colors help differentiate prompts and user input.
- Represents the central hub of navigation.

4.4 View Inventory

Function: Lists all stock items and their quantities.



- Shows all seven electronics items in a structured table format.
- Columns include ID, Item Name, and Quantity.
- Items with quantity ≤ 3 are highlighted in blinking red using ANSI escape sequences.
- Makes low-stock warnings instantly noticeable.

4.5 Add/Restock Item

Function: Updates inventory by increasing the stock of a selected item.

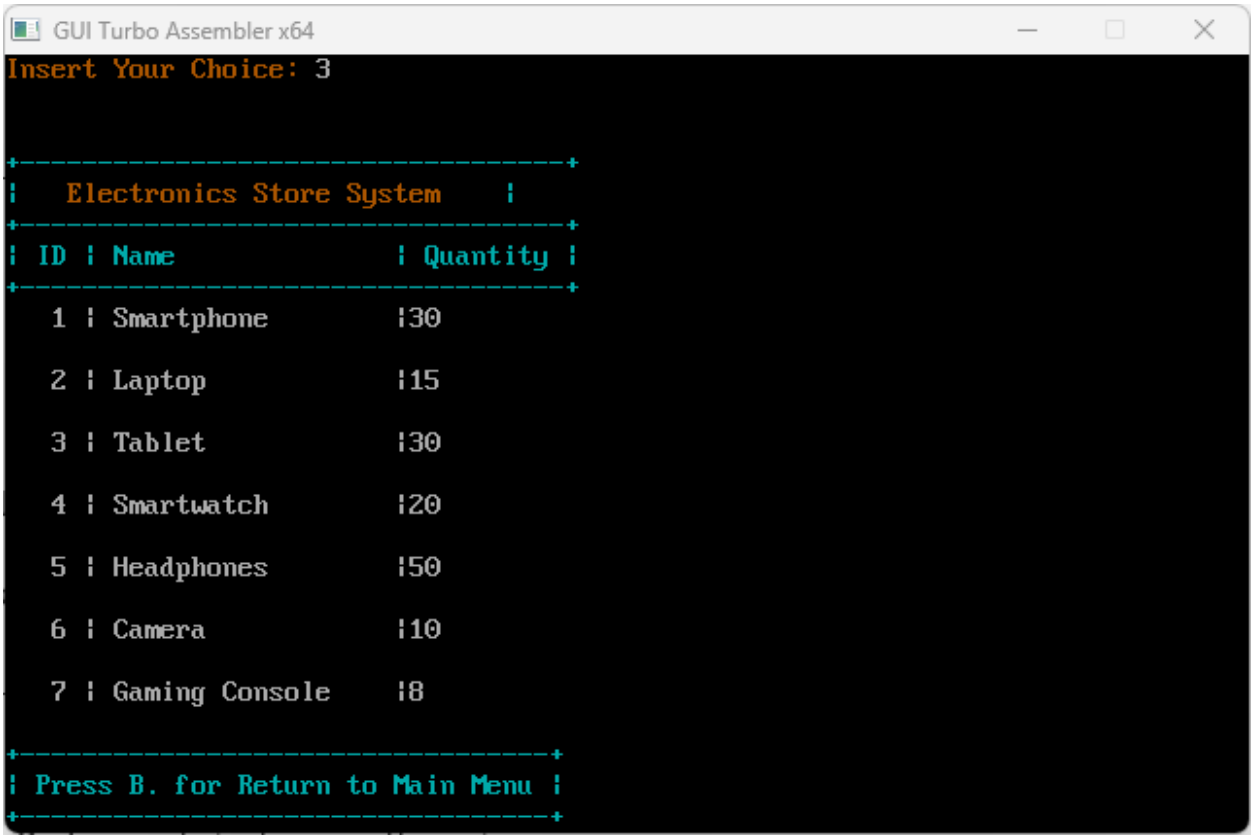
```
GUI Turbo Assembler x64
Insert Your Choice: 2


+-----------------------------------+
¦   Electronics Store System     ¦
+-----------------------------------+
¦ ID ¦ Name               ¦ Quantity ¦
+-----------------------------------+
  1 ¦ Smartphone         ¦25

  2 ¦ Laptop             ¦15

  3 ¦ Tablet             ¦30

  4 ¦ Smartwatch         ¦20

  5 ¦ Headphones         ¦50

  6 ¦ Camera             ¦10

  7 ¦ Gaming Console     ¦8

+-----------------------------------+
¦ Press B. for Return to Main Menu ¦
+-----------------------------------+
```

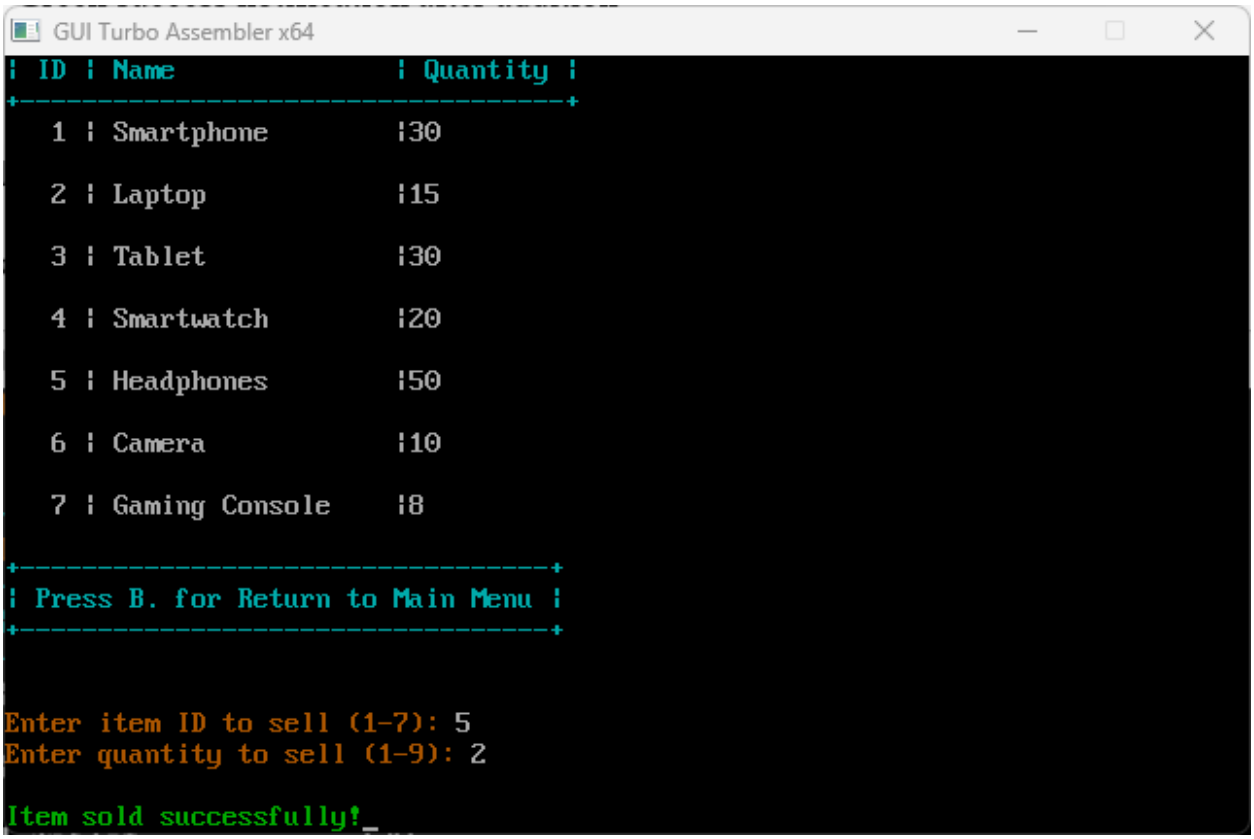
```
GUI Turbo Assembler x64
¦ ID ¦ Name               ¦ Quantity ¦
+-----------------------------------+
  1 ¦ Smartphone         ¦25

  2 ¦ Laptop             ¦15

  3 ¦ Tablet             ¦30

  4 ¦ Smartwatch         ¦20

  5 ¦ Headphones         ¦50

  6 ¦ Camera             ¦10

  7 ¦ Gaming Console     ¦8

+-----------------------------------+
¦ Press B. for Return to Main Menu ¦
+-----------------------------------+


Enter item ID to restock (1-7): 1
Enter quantity to restock (1-9): 5

Item restocked successfully!_
```

- Prompts user for an item ID and quantity to restock.
- Validates input and updates the corresponding inventory count.
- Confirms successful restocking with a green-colored message.
- Smooth transition back to the main interface ensures user flow is maintained.

4.6 Sell Item

Function: Processes an item sale and adjusts the inventory and sales log.

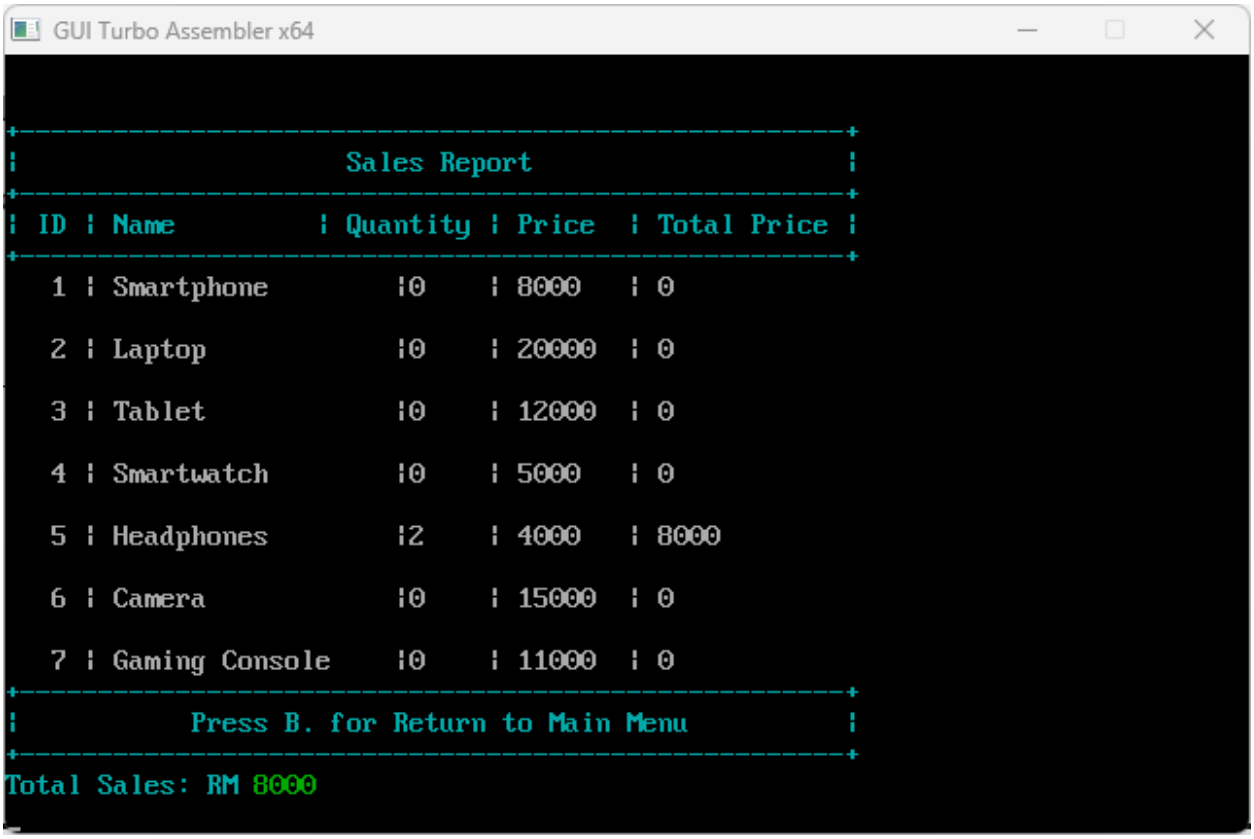




- Prompts the user to enter an item ID and quantity to sell.
- Performs error checks for insufficient stock.
- On success, updates the stock level, total sales, and displays a success message.

4.7 Sales Report

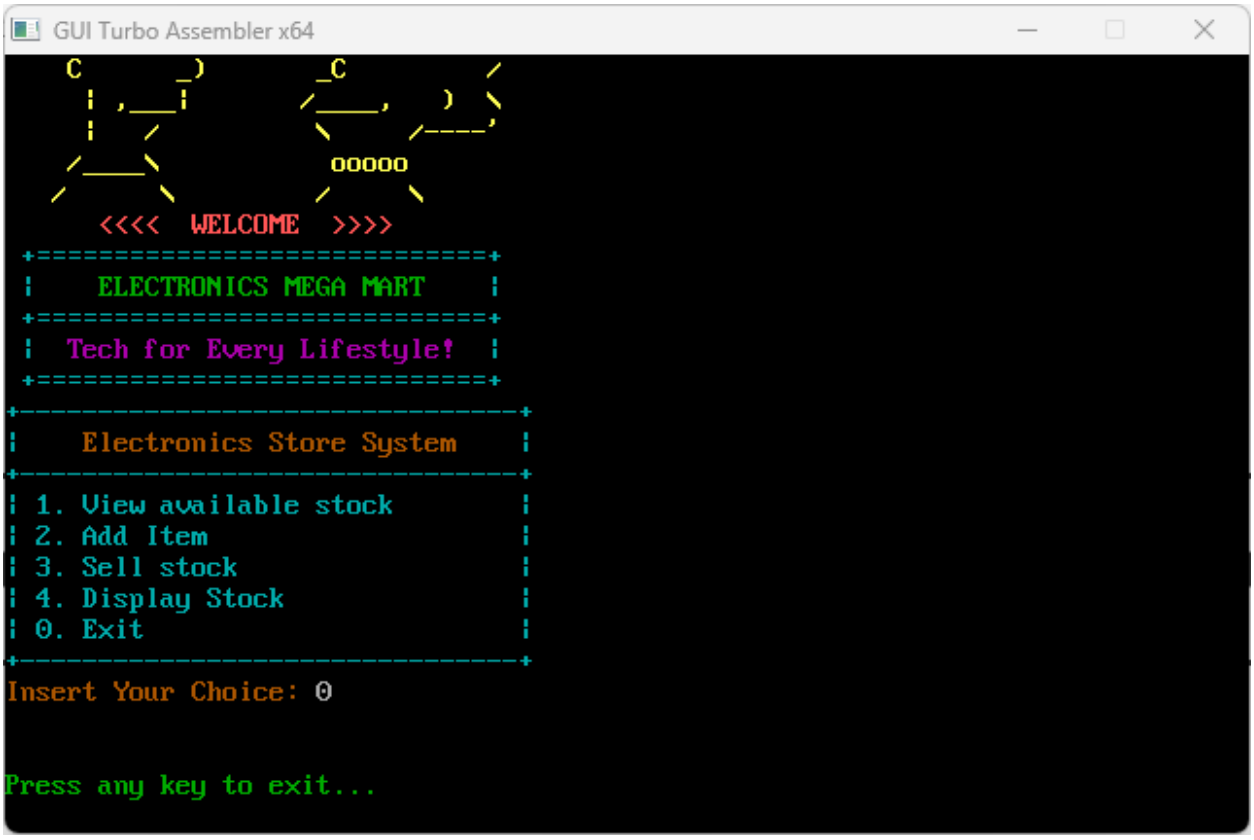
Function: Displays a structured report of total item sales.



- Displays a formatted table of each item's name, quantity sold, price per unit, and total sales per item.
- Concludes with the grand total value across all items.
- Highlights total earnings in green to indicate revenue summary.
- Ensures that report data aligns with actual transaction history.

4.8 Exit Screen

Function: Closes the system with a final thank-you message.



- Activated when the user selects option '0'.
- System calls DOS interrupt int 21h with function 4Ch to terminate the program.
- Demonstrates a clean and professional exit from the application.
- Ensure all output buffers are flushed and memory is released.

5.0: Source Code with Explanation

This section provides an overview of the key components in the Assembly language source code for the Electronics Store Inventory System. The explanation is grouped by the major segments and procedures used in the .ASM file.

5.1 .DATA Segment Explanation

The .DATA section is where all initialized data values, constant messages, color codes, inventory, prices, counters, and interface assets like ASCII art are declared. Here's how each group is used

5.1.1 Color Definitions

```
.DATA
    ; Color definitions
    redBlinkStart db 27, '[5;31m$', 0    ; Blink and red text
    redBlinkEnd db 27, '[0m$', 0         ; Reset color
    greenText db 27, '[32m$', 0          ; Green text
    blueText db 27, '[34m$', 0           ; Blue text
    magentaText db 27, '[35m$', 0        ; Magenta text
    cyanText db 27, '[36m$', 0           ; Cyan text
    whiteText db 27, '[37m$', 0          ; White text
    resetColor db 27, '[0m$', 0          ; Reset color
    OrangeText db 27, '[33m$', 0         ; Orange text
    brightYellowText db 27, '[1;33m$', 0; Bright Yellow text
```

A set of ANSI escape codes is used to apply colored and blinking text output to the console. These are used for UI emphasis (e.g., blinking red for warnings, green for success):

- redBlinkStart, redBlinkEnd — for blinking alerts
- greenText, blueText, magentaText, cyanText, yellowText, whiteText — used to color different text areas
- resetColor — to return text to default formatting

5.1.2 Login and Greeting System

```
; Login system
loginHeader db 27, '[36m', 10,10, '+--------------------------------+'
                           db 10, '|     Electronics Store Login    |'
                           db 10, '+--------------------------------+', 27, '[0m$'
askUsername db 27, '[33m', 10,10, 'Enter Your User Name: ', 27, '[32m$'
greetingMessage db 27, '[36m', 13,10,10, 'Welcome To Electronics Store System  ', 27, '[0m$'
buffer db 100 dup(0)
```

- loginHeader, askUsername, greetingMessage, and buffer are used to interact with the user during login and greeting.
- buffer stores the input username captured via DOS interrupt 21h (function 0Ah).

5.1.3 ASCII Art and Branding

```
; ASCII Art Electronics with colors
electronicsArt db 27, '[1;33m',10
            ; Left Side Robot
            db '     |\/\/\/|         /\ /\ /\',10
            db '     |      |        | V  \/  \---.',10
            db '     |      |         \_         /',10
            db '     | (o)(o)          (o)(o)  <__.',10
            db '     C       _)        _C         /',10
            db '      | ,___|         /____,    )  \',10
            db '      |   /            \     /----''',10
            db '     /____\             ooooo',10
            db '    /     \            /     \',10
            db 27, '[36m'
            db '', 27, '[5;31m', '      <<<<  WELCOME  >>>>', 27, '[0;36m',10
            db ' +==============================+',10
            db ' |    ', 27, '[5;32m', 'ELECTRONICS MEGA MART', 27, '[0;36m', '    |',10
            db ' +==============================+',10
            db ' |  ', 27, '[5;35m', 'Tech for Every Lifestyle!', 27, '[0;36m', '  |',10
            db ' +==============================+',10
            db 27, '[0m$'
```

- electronicsArt is a multi-line art graphic created using printable characters and color codes to enhance the appearance of the system. It includes two symmetrical robots and a centered welcome banner with a slogan for visual appeal.

5.1.4 Menu and Error Messages

```
mainMenu db 27, '[36m', "+--------------------------------+"
                 db 10, "|    ", 27, '[33m', "Electronics Store System", 27, '[36m', "    |"
                 db 10, "+--------------------------------+"
                 db 10, "| 1. View available stock        |"
                 db 10, "| 2. Add Item                    |"
                 db 10, "| 3. Sell stock                  |"
                 db 10, "| 4. Display Stock               |"
                 db 10, "| 0. Exit                        |"
                 db 10, "+--------------------------------+"
          db 10, 27, '[33m', "Insert Your Choice: ", 27, '[32m$'

invalidInput db 27, '[31m', 10, 10, "+--------------------------------+"
                            db 10, "|             Error!             |"
                            db 10, "| Please enter a proper number!  |"
                            db 10, "+--------------------------------+", 27, '[0m$'
```

- mainMenu displays the core options the user can choose from.
- invalidInput is triggered when the user provides an incorrect menu option.

5.1.5 Inventory Variables

```
; Initial Stock Values
smartphoneStock dw 25
laptopStock dw 15
tabletStock dw 30
smartwatchStock dw 20
headphonesStock dw 50
cameraStock dw 10
gamingConsoleStock dw 8
```

- Each item category such as smartphoneStock, laptopStock, etc., stores the current quantity for that product.
- These are modified during the restock and sell processes.

5.1.6 Price Definitions

```
; Prices (split into high and low parts)
smartphonePriceLow dw 1F40h     ; 8000 = 1F40h
smartphonePriceHigh dw 0h
laptopPriceLow dw 4E20h         ; 20000 = 4E20h
laptopPriceHigh dw 0h
tabletPriceLow dw 2EE0h         ; 12000 = 2EE0h
tabletPriceHigh dw 0h
smartwatchPriceLow dw 1388h     ; 5000 = 1388h
smartwatchPriceHigh dw 0h
headphonesPriceLow dw 0FA0h     ; 4000 = 0FA0h
headphonesPriceHigh dw 0h
cameraPriceLow dw 3A98h         ; 15000 = 3A98h
cameraPriceHigh dw 0h
gamingConsolePriceLow dw 2AF8h ; 11000 = 2AF8h
gamingConsolePriceHigh dw 0h
```

- Each item has an associated low-word price value (e.g., smartphonePriceLow) stored in cents (e.g., 8000 = RM80.00).
- The prices are used during sales calculations.

5.1.7 Sales Counters

```
; Sales and Cart
smartphoneSold dw 0
laptopSold dw 0
tabletSold dw 0
smartwatchSold dw 0
headphonesSold dw 0
cameraSold dw 0
gamingConsoleSold dw 0
totalSales dw 0    ; Total sales
```

- Variables like smartphoneSold, laptopSold, etc., track the number of units sold.
- totalSales accumulates the overall earnings in cents.

5.1.8 Prompts and Messages

```
restockPrompt db 27, '[33m', 10,10, "Enter item ID to restock (1-7): ", 27, '[32m$'
restockQtyPrompt db 27, '[33m', 10, "Enter quantity to restock (1-9): ", 27, '[32m$'
restockSuccess db 27, '[32m', 10,10, "Item restocked successfully!", 27, '[0m$'

sellPrompt db 27, '[33m', 10,10, "Enter item ID to sell (1-7): ", 27, '[32m$'
sellQtyPrompt db 27, '[33m', 10, "Enter quantity to sell (1-9): ", 27, '[32m$'
sellSuccess db 27, '[32m', 10,10, "Item sold successfully!", 27, '[0m$'
notEnoughStock db 27, '[31m', 10,10, "Not enough stock!", 27, '[0m$'

totalSalesHeader db 27, '[36m', 10,10,10, "Total Sales: RM", 27, '[32m$'
```

- These include system instructions such as restockPrompt, sellPrompt, sellSuccess, and notEnoughStock, which guide the user through system operations.

5.1.9 Inventory and Sales Layout

```
; Inventory
inventoryHeader db 27, '[36m', 10,10,10, "+-----------------------------------+"
                                         db 10, "|   ", 27, '[33m', "Electronics Store System", 27, '[36m', "    |"
                                         db 10, "+-----------------------------------+"
                                         db 10, "| ID | Name              | Quantity |"
                                         db 10, "+-----------------------------------+", 27, '[0m$'

  smartphone          db 27, '[37m', 10, "   1 | Smartphone        |", 27, '[0m$'
  laptop              db 27, '[37m', 10, "   2 | Laptop            |", 27, '[0m$'
  tablet              db 27, '[37m', 10, "   3 | Tablet            |", 27, '[0m$'
  smartwatch          db 27, '[37m', 10, "   4 | Smartwatch        |", 27, '[0m$'
  headphones          db 27, '[37m', 10, "   5 | Headphones        |", 27, '[0m$'
  camera              db 27, '[37m', 10, "   6 | Camera            |", 27, '[0m$'
  gamingConsole       db 27, '[37m', 10, "   7 | Gaming Console    |", 27, '[0m$'


; Sales Report
salesOrderHeader  db 27, '[36m', 10,10,10, "+-------------------------------------------------------+"
                                         db 10, "|                      Sales Report                     |"
                                         db 10, "+-------------------------------------------------------+"
                                         db 10, "| ID | Name          | Quantity | Price  | Total Price |"
                                         db 10, "+-------------------------------------------------------+", 27, '[0m$'

backBttn db 27, '[36m', 10, "| Press B. for Return to Main Menu |"
                     db 10, "+----------------------------------+", 27, '[0m$'

bottomBorder db 27, '[36m', 10, "+----------------------------------+", 27, '[0m$'
totalSalesFooter db 27, '[36m', 10, "Total Sales: RM ", 27, '[32m$'
```

```
lowStock db 27, '[31m', 'LOW STOCK', 27, '[0m$'

; Sales Report specific strings
salesReportBackBttn     db 27, '[36m', 10, "|           Press B. for Return to Main Menu          |"
                                   db 10, "+-----------------------------------------------------+", 27, '[0m$'
salesReportBottomBorder db 27, '[36m', 10, "+-----------------------------------------------------+", 27, '[0m$'
```

- Variables like inventoryHeader, smartphone, tablet, etc., store formatted strings used to render a visual table of stock.
- salesOrderHeader, salesReportBackBttn, and related entries are used in the report interface.

5.1.10 Formatting Helpers

```
endline db 10,'$'
newline db 10, '$'
separator  db '     | $'
separator1 db '    | $'
separator2 db '   | $'
```

- endline, newline, separator, etc., are used to ensure neat and consistent output formatting.

5.2 CODE Segment Explanation

The CODE segment contains executable logic for the system. It is divided into labeled procedures (PROC) for modular functionality, allowing for clarity, reuse, and structured control flow.

The CODE section starts with the setup of the program structure and then outlines the key procedures

```
.MODEL SMALL
.STACK 100H
.386
```

5.2.1 Main PROC

```
Main PROC
    ; Initialize data segment
    mov ax, @data
    mov ds, ax

    ; Display login header
    mov ah, 09h
    lea dx, loginHeader
    int 21h

    ; Ask user's username
    mov ah, 09h
    lea dx, askUsername
    int 21h

    ; Read String (Not Character)
    mov buffer[0], 21
    mov ah, 0Ah
    lea dx, buffer
    int 21h

    ; Display greeting
    mov ah, 09h
    lea dx, greetingMessage
    int 21h

    mov bx, 2
    add bl, buffer[1]
    mov buffer[bx], '$'

    mov ah, 09h
    lea dx, buffer
    add dx, 2
    int 21h

    jmp MainInterface
```

- Initializes the DS register to access .DATA.
- Displays the login screen and accepts the user's name.
- Greetings the user using their input.
- Jumps to MainInterface for further interaction.

5.2.2 Main Interface

```
MainInterface:
    ; Display ASCII Art Electronics
    mov ah, 09h
    lea dx, electronicsArt
    int 21h

    ; Display Main Menu
    mov ah, 09h
    lea dx, mainMenu
    int 21h

    ; Character Input
    mov ah, 01h
    int 21h

    ; Conditions based on input
    cmp al, '1'
    je viewInventory

    cmp al, '2'
    je restockItem

    cmp al, '3'
    je sellItems

    cmp al, '4'
    je salesReport

    cmp al, '0'
    je exit

    jmp error
```

- Serves as the main control center.
- Displays ASCII art and the interactive menu.
- Handles user input and routes to corresponding functions.
- Handles error input with a red warning box.

5.2.3 view Inventory

```
viewInventory:
    call ShowInventory
    jmp MainInterface
```

- Prints a formatted inventory table.
- Uses DisplayIntegerHighlight to highlight low-stock items (≤ 3) with blinking red.
- Prompts user to return to the menu by pressing 'B'.

5.2.4 restock Item

```
restockItem:
    ; Display inventory
    call ShowInventory

    ; Prompt for item ID to restock
    mov ah, 09h
    lea dx, restockPrompt
    int 21h
    mov ah, 01h
    int 21h
    sub al, '0'
    mov bl, al

    ; Prompt for quantity to restock
    mov ah, 09h
    lea dx, restockQtyPrompt
    int 21h
    mov ah, 01h
    int 21h
    sub al, '0'
    mov cl, al

    ; Find and restock item
    cmp bl, 1
    je restockSmartphone
    cmp bl, 2
    je restockLaptop
    cmp bl, 3
    je restockTablet
    cmp bl, 4
    je restockSmartwatch
    cmp bl, 5
    je restockHeadphones
    cmp bl, 6
    je restockCamera
    cmp bl, 7
    je restockGamingConsole
    jmp error
```

```
restockSmartphone:
    mov ax, smartphoneStock
    add ax, cx
    mov smartphoneStock, ax
    jmp restockSuccessLabel

restockLaptop:
    mov ax, laptopStock
    add ax, cx
    mov laptopStock, ax
    jmp restockSuccessLabel

restockTablet:
    mov ax, tabletStock
    add ax, cx
    mov tabletStock, ax
    jmp restockSuccessLabel

restockSmartwatch:
    mov ax, smartwatchStock
    add ax, cx
    mov smartwatchStock, ax
    jmp restockSuccessLabel

restockHeadphones:
    mov ax, headphonesStock
    add ax, cx
    mov headphonesStock, ax
    jmp restockSuccessLabel

restockCamera:
    mov ax, cameraStock
    add ax, cx
    mov cameraStock, ax
    jmp restockSuccessLabel

restockGamingConsole:
    mov ax, gamingConsoleStock
    add ax, cx
    mov gamingConsoleStock, ax
    jmp restockSuccessLabel

restockSuccessLabel:
    mov ah, 09h
    lea dx, restockSuccess
    int 21h
    mov ah, 01h
    int 21h
    jmp MainInterface
```

- Prompts the user for an item ID and quantity.
- Uses separate labels like restockSmartphone, restockLaptop, etc. to update stock for each item.
- Displays confirmation message after update.

5.2.5 sell Items

```
sellItems:
    ; Display inventory
    call ShowInventory

    ; Prompt for item ID to sell
    mov ah, 09h
    lea dx, sellPrompt
    int 21h
    mov ah, 01h
    int 21h
    sub al, '0'
    mov bl, al

    ; Prompt for quantity to sell
    mov ah, 09h
    lea dx, sellQtyPrompt
    int 21h
    mov ah, 01h
    int 21h
    sub al, '0'
    mov cl, al

    ; Find and sell item
    cmp bl, 1
    je sellSmartphone
    cmp bl, 2
    je sellLaptop
    cmp bl, 3
    je sellTablet
    cmp bl, 4
    je sellSmartwatch
    cmp bl, 5
    je sellHeadphones
    cmp bl, 6
    je sellCamera
    cmp bl, 7
    je sellGamingConsole
    jmp error

sellSmartphone:
    mov ax, smartphoneStock
    cmp ax, cx
    jb notEnoughStockLabel
    sub ax, cx
    mov smartphoneStock, ax
    mov ax, cx
    mov bx, smartphonePriceLow
    imul bx
    add totalSales, ax
    add smartphoneSold, cx
    jmp sellSuccessLabel

sellLaptop:
    mov ax, laptopStock
    cmp ax, cx
    jb notEnoughStockLabel
    sub ax, cx
    mov laptopStock, ax
    mov ax, cx
    mov bx, laptopPriceLow
    imul bx
    add totalSales, ax
    add laptopSold, cx
    jmp sellSuccessLabel

sellTablet:
    mov ax, tabletStock
    cmp ax, cx
    jb notEnoughStockLabel
    sub ax, cx
    mov tabletStock, ax
    mov ax, cx
    mov bx, tabletPriceLow
    imul bx
    add totalSales, ax
    add tabletSold, cx
    jmp sellSuccessLabel

sellSmartwatch:
    mov ax, smartwatchStock
    cmp ax, cx
    jb notEnoughStockLabel
    sub ax, cx
    mov smartwatchStock, ax
    mov ax, cx
    mov bx, smartwatchPriceLow
    imul bx
    add totalSales, ax
    add smartwatchSold, cx
    jmp sellSuccessLabel

sellHeadphones:
    mov ax, headphonesStock
    cmp ax, cx
    jb notEnoughStockLabel
    sub ax, cx
    mov headphonesStock, ax
    mov ax, cx
    mov bx, headphonesPriceLow
    imul bx
    add totalSales, ax
    add headphonesSold, cx
    jmp sellSuccessLabel

sellCamera:
    mov ax, cameraStock
    cmp ax, cx
    jb notEnoughStockLabel
    sub ax, cx
    mov cameraStock, ax
    mov ax, cx
    mov bx, cameraPriceLow
    imul bx
    add totalSales, ax
    add cameraSold, cx
    jmp sellSuccessLabel

sellGamingConsole:
    mov ax, gamingConsoleStock
    cmp ax, cx
    jb notEnoughStockLabel
    sub ax, cx
    mov gamingConsoleStock, ax
    mov ax, cx
    mov bx, gamingConsolePriceLow
    imul bx
    add totalSales, ax
    add gamingConsoleSold, cx
    jmp sellSuccessLabel

sellSuccessLabel:
    mov ah, 09h
    lea dx, sellSuccess
    int 21h
    mov ah, 01h
    int 21h
    jmp MainInterface
```

- Prompts for item ID and quantity to sell.
- Validates stock availability; triggers an error if insufficient.
- Deducts quantity from stock, adds to sales count and totalSales.
- Displays either success or error message.

5.2.6 Show Sales Report

```
ShowSalesReport PROC
    ; Display sales report header
    mov ah, 09h
    lea dx, salesOrderHeader
    int 21h

    ; Display smartphone sales
    mov ah, 09h
    lea dx, smartphone
    int 21h
    mov ax, smartphoneSold
    call DisplayInteger
    mov ah, 09h
    lea dx, separator
    int 21h
    mov ax, smartphonePriceLow
    call DisplayInteger
    mov ah, 09h
    lea dx, separator1
    int 21h
    mov ax, smartphoneSold
    mov bx, smartphonePriceLow
    imul bx
    call DisplayInteger
    mov ah, 09h
    lea dx, endline
    int 21h

    ; Display laptop sales
    call DisplaySalesItem
    mov ah, 09h
    lea dx, laptop
    int 21h
    mov ax, laptopSold
    call DisplayInteger
    mov ah, 09h
    lea dx, separator
    int 21h
    mov ax, laptopPriceLow
    call DisplayInteger
    mov ah, 09h
    lea dx, separator2
    int 21h
    mov ax, laptopSold
    mov bx, laptopPriceLow
    imul bx
    call DisplayInteger
    mov ah, 09h
    lea dx, endline
    int 21h

    ; Display tablet sales
    call DisplaySalesItem
    mov ah, 09h
    lea dx, tablet
    int 21h
    mov ax, tabletSold
    call DisplayInteger
    mov ah, 09h
    lea dx, separator
    int 21h
    mov ax, tabletPriceLow
    call DisplayInteger
    mov ah, 09h
    lea dx, separator2
    int 21h
    mov ax, tabletSold
    mov bx, tabletPriceLow
    imul bx
    call DisplayInteger
    mov ah, 09h
    lea dx, endline
    int 21h

    ; Display smartwatch sales
    call DisplaySalesItem
    mov ah, 09h
    lea dx, smartwatch
    int 21h
    mov ax, smartwatchSold
    call DisplayInteger
    mov ah, 09h
    lea dx, separator
    int 21h
    mov ax, smartwatchPriceLow
    call DisplayInteger
    mov ah, 09h
    lea dx, separator1
    int 21h
    mov ax, smartwatchSold
    mov bx, smartwatchPriceLow
    imul bx
    call DisplayInteger
    mov ah, 09h
    lea dx, endline
    int 21h

    ; Display headphones sales
    call DisplaySalesItem
    mov ah, 09h
    lea dx, headphones
    int 21h
    mov ax, headphonesSold
    call DisplayInteger
    mov ah, 09h
    lea dx, separator
    int 21h
    mov ax, headphonesPriceLow
    call DisplayInteger
    mov ah, 09h
    lea dx, separator1
    int 21h
    mov ax, headphonesSold
    mov bx, headphonesPriceLow
    imul bx
    call DisplayInteger
    mov ah, 09h
    lea dx, endline
    int 21h

    ; Display camera sales
    call DisplaySalesItem
    mov ah, 09h
    lea dx, camera
    int 21h
    mov ax, cameraSold
    call DisplayInteger
    mov ah, 09h
    lea dx, separator
    int 21h
    mov ax, cameraPriceLow
    call DisplayInteger
    mov ah, 09h
    lea dx, separator2
    int 21h
    mov ax, cameraSold
    mov bx, cameraPriceLow
    imul bx
    call DisplayInteger
    mov ah, 09h
    lea dx, endline
    int 21h

    ; Display gaming console sales
    call DisplaySalesItem
    mov ah, 09h
    lea dx, gamingConsole
    int 21h
    mov ax, gamingConsoleSold
    call DisplayInteger
    mov ah, 09h
    lea dx, separator
    int 21h
    mov ax, gamingConsolePriceLow
    call DisplayInteger
    mov ah, 09h
    lea dx, separator2
    int 21h
    mov ax, gamingConsoleSold
    mov bx, gamingConsolePriceLow
    imul bx
    call DisplayInteger
    mov ah, 09h
    int 21h

    ; Display return to main menu button
    mov ah, 09h
    lea dx, salesReportBottomBorder
    int 21h
    mov ah, 09h
    lea dx, salesReportBackBttn
    int 21h

    ; Display total sales
    mov ah, 09h
    lea dx, totalSalesFooter
    int 21h
    mov ax, totalSales
    call DisplayInteger
    mov ah, 09h
    lea dx, newline
    int 21h

    ; Wait for user input
    mov ah, 01h
    int 21h
    cmp al, 'B'
    je MainInterface
    cmp al, 'b'
    je MainInterface

    ret
ShowSalesReport ENDP
```

- Lists all products with columns for quantity sold, unit price, and total value.
- Calculates total sales using multiplication (imul) and displays using DisplayInteger.
- Uses salesReportBackBttn to prompt user to return.

5.2.7 Show Inventory

```
ShowInventory PROC
    ; Display inventory
    mov ah, 09h
    lea dx, inventoryHeader
    int 21h

    ; Display smartphone stock
    mov ah, 09h
    lea dx, smartphone
    int 21h
    mov ax, smartphoneStock
    call DisplayIntegerHighlight
    mov ah, 09h
    lea dx, newline
    int 21h

    ; Display laptop stock
    mov ah, 09h
    lea dx, laptop
    int 21h
    mov ax, laptopStock
    call DisplayIntegerHighlight
    mov ah, 09h
    lea dx, newline
    int 21h

    ; Display tablet stock
    mov ah, 09h
    lea dx, tablet
    int 21h
    mov ax, tabletStock
    call DisplayIntegerHighlight
    mov ah, 09h
    lea dx, newline
    int 21h

    ; Display smartwatch stock
    mov ah, 09h
    lea dx, smartwatch
    int 21h
    mov ax, smartwatchStock
    call DisplayIntegerHighlight
    mov ah, 09h
    lea dx, newline
    int 21h

    ; Display headphones stock
    mov ah, 09h
    lea dx, headphones
    int 21h
    mov ax, headphonesStock
    call DisplayIntegerHighlight
    mov ah, 09h
    lea dx, newline
    int 21h

    ; Display camera stock
    mov ah, 09h
    lea dx, camera
    int 21h
    mov ax, cameraStock
    call DisplayIntegerHighlight
    mov ah, 09h
    lea dx, newline
    int 21h

    ; Display gaming console stock
    mov ah, 09h
    lea dx, gamingConsole
    int 21h
    mov ax, gamingConsoleStock
    call DisplayIntegerHighlight
    mov ah, 09h
    lea dx, newline
    int 21h

    ; Display bottom border
    mov ah, 09h
    lea dx, bottomBorder
    int 21h

    ; Display return to main menu option
    mov ah, 09h
    lea dx, backBttn
    int 21h

    ; Wait for user input
    mov ah, 01h
    int 21h
    cmp al, 'B'
    je MainInterface
    cmp al, 'b'
    je MainInterface

    ret
ShowInventory ENDP
```

- Lists each item's stock value in an organized table format.

- Displays color-coded warnings for low inventory.

5.2.8 Display Integer Highlight

```
DisplayIntegerHighlight PROC
    push ax
    push bx
    push cx
    push dx

    ; Save the original quantity value
    mov bx, ax

    ; Check if original quantity is less than or equal to 3
    cmp bx, 3
    jle highlight_loop

    ; Convert integer to string for normal display
    mov ax, bx
    mov cx, 0
    mov dx, 0
    mov bx, 10

convert_loop_normal:
    xor dx, dx
    div bx
    add dl, '0'
    push dx
    inc cx
    test ax, ax
    jnz convert_loop_normal

NormalDisplay:
    ; Print the number normally
    print_loop_normal:
        pop dx
        mov ah, 02h
        int 21h
        loop print_loop_normal

    jmp ContinueExecution

highlight_loop:
    ; Start red blinking text
    mov ah, 09h
    lea dx, redBlinkStart
    int 21h

    ; Convert integer to string for highlight display
    mov ax, bx
    mov cx, 0
    mov dx, 0
    mov bx, 10

convert_loop_highlight:
    xor dx, dx
    div bx
    add dl, '0'
    push dx
    inc cx
    test ax, ax
    jnz convert_loop_highlight

    ; Print the digits with highlight
    highlight:
        pop dx
        mov ah, 02h
        int 21h
        loop highlight

    ; End red blinking text
    mov ah, 09h
    lea dx, redBlinkEnd
    int 21h

    jmp ContinueExecution

ContinueExecution:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
DisplayIntegerHighlight ENDP
```

- Determines if value ≤ 3.
- If true, wraps it in red blinking text.
- If false, prints it in standard white.
- Uses division by 10 to extract digits for output.

5.2.9 Display Integer

```
DisplayInteger PROC
    ; Convert integer to string and display
    mov cx, 0
    mov bx, 10

convertToString:
    mov dx, 0
    div bx
    push ax
    add dl, '0'
    pop ax
    push dx
    inc cx
    cmp ax, 0
    jnz convertToString

    mov ah, 02h

displayString:
    pop dx
    int 21h
    dec cx
    jnz displayString

    ret
DisplayInteger ENDP
```

- Converts numeric values in AX to their ASCII representations.
- Pushes digits in reverse and prints via popping stack.

5.2.10 Display Sales Item

```
DisplaySalesItem PROC
    ret
DisplaySalesItem ENDP
```

- Currently a placeholder for future modular formatting.
- Ensures consistent output style across all item rows.

6.0 Conclusion

This project has demonstrated the practical application of assembly language in building a functional Electronics Store Inventory System. By utilizing x86 assembly, the system effectively manages product stocks, handles restocking and sales, and generates a formatted sales report. The use of low-level programming highlights the level of control and efficiency achievable in hardware-near operations, a key requirement in systems where performance and memory optimization are critical.

Through this development, the project reinforced the importance of understanding processor architecture, memory management, and interrupt-driven I/O in system-level programming. The implementation also illustrates how assembly language remains relevant in domains such as cybersecurity and digital forensics, where transparency, traceability, and minimal abstraction layers are vital.

Overall, this system reflects a solid grasp of low-level programming concepts, showcasing how assembly can still be employed in modern educational and specialized applications requiring full control of system behavior.

References

Casey, E. (2011). *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet*. Academic Press.

Eilam, E. (2011). *Reversing: Secrets of Reverse Engineering*. Wiley.

Ligh, M. H., Adair, S., Hartstein, B., & Richard, M. (2014). *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*. Wiley.

Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press.