[OVERVIEW]

The Library Management System is a software application designed to streamline the management of library resources, including books, members, and transactions. The system will enable users to perform various functions such as adding and removing books, checking in and checking out items, and maintaining member records. By using assembly language, the project focuses on low-level programming techniques, optimizing performance and memory usage.

[ABOUT PROJECT]

This project utilizes the Irvine32 library to simplify tasks such as input/output operations, string handling, and memory management. The library management system will include a user-friendly interface that allows librarians to easily navigate through options. Key features will include:

* Access/Log in methodologies into system for Librarians
* Adding new books and members
* Searching for books and members
* Checking out and returning books
* Displaying current inventory and member details
* Handling requisitions for books for specific intervals
* Update Book information (regarding name, author, title or quantity)
* Menu driven interface separate for both customer/client and a librarian/user of the management system.

[SOFTWARE APPROACH]

The software will be structured using a modular approach, dividing functionality into distinct sections (modules) for clarity and maintainability. Each module will handle specific tasks, such as:

* Data storage and retrieval (using arrays or records)
* User interface management (menu-driven interactions)
* Input validation and error handling Irvine32.inc leveraged for its powerful macros and procedures, allowing for efficient development of assembly code.
* Irvine32.inc library integrate built-in functionalities and processing syntaxes for in main program code hierarchy.

[PROGRAM COMPOSITION]

.DATA SEGMENT

|  |  |  |
| --- | --- | --- |
| VARIABLE NAME | DATA TYPE | VALUE / DESCRIPTION |
| WELCOMEMESSAGE | BYTE | "---------------------------------------", 0 |
| WELCOMEMESSAGE2 | BYTE | " |
| WELCOMEMESSAGE3 | BYTE | " |
| WELCOMEMESSAGE4 | BYTE | " |
| WELCOMEMESSAGE5 | BYTE | "---------------------------------------", 0 |
| EMPTYSTRING | BYTE | " ", 0 |
| DISPAY1 | BYTE | Multi-line display for "3" character |
| DISPAY2 | BYTE | Multi-line display for "2" character |
| DISPAY3 | BYTE | Multi-line display for "1" character |
| LINE1 | BYTE | Welcome message design pattern |
| EXITMESSAGE | BYTE | "Thank you for using the program. Goodbye!", 0 |
| BOOKTITLEMSG | BYTE | "Book Title: ", 0 |
| BOOKAUTHORMSG | BYTE | "Book Author: ", 0 |
| BOOKISBNMSG | BYTE | "Book ISBN: ", 0 |
| BOOKNOTFOUND | BYTE | "Book not found.", 0 |
| NUMBER\_OF\_BOOKS | DWORD | 50 |
| BOOKRETURNMSG | BYTE | "Book Returned Successfully", 0 |
| BOOKFAILMSG | BYTE | "No Book Rented yet", 0 |
| IDPROMPT | BYTE | "Enter your Employee ID: ", 0 |
| NAMEPROMPT | BYTE | "Enter your Name: ", 0 |
| YEARPROMPT | BYTE | "Enter your Year of Birth: ", 0 |
| BOOKFOUND | BYTE | "Foudn!", 0 |
| EMPLOYEEID | DWORD | ? (Uninitialized) |
| EMPLOYEENAME | BYTE | 50 DUP(0) (Array for 50 characters) |
| YEAROFBIRTH | DWORD | ? (Uninitialized) |
| USERTYPEPROMPT | BYTE | "Are you a: (1) Librarian (2) Customer: ", 0 |
| LIBRARIANOPTIONS | BYTE | Librarian options menu string |
| CUSTOMEROPTIONS | BYTE | Customer options menu string |
| CURRENTID | DWORD | 0 |
| BOOKTITLEPROMPT | BYTE | "Enter Book Title: ", 0 |
| BOOKAUTHORPROMPT | BYTE | "Enter Book Author: ", 0 |
| BOOKISBNPROMPT | BYTE | "Enter Book ISBN: ", 0 |
| QUANTITYPROMPT | BYTE | "Enter Book Quantity: ", 0 |
| BOOKADDEDMESSAGE | BYTE | "Book successfully added!", 0 |
| BOOKNOTFOUNDMESSAGE | BYTE | "Book not found.", 0 |
| UPDATETITLEPROMPT | BYTE | "Enter new Book Title: ", 0 |
| UPDATEAUTHORPROMPT | BYTE | "Enter new Book Author: ", 0 |
| QUANTITYUPDATEDPROMPT | BYTE | "Enter new Quantity", 0 |
| BOOKUPDATEDMESSAGE | BYTE | "Book information updated.", 0 |
| BOOKRENT | DWORD | 5h |
| BOOKRENTMSG | BYTE | "Enter the book you want to rent", 0 |
| F1 | BYTE | "Book Found at this index", 0 |
| F2 | BYTE | "Book not found", 0 |
| MAX\_BOOKS | EQU | 50 |
| BOOKTITLES | BYTE | MAX\_BOOKS \* BookTitleSize DUP(0) |
| BOOKAUTHORS | BYTE | MAX\_BOOKS \* BookAuthorSize DUP(0) |
| BOOKISBNS | DWORD | MAX\_BOOKS DUP(? ) |
| QUANTITY | DWORD | MAX\_BOOKS DUP(? ) |
| BOOKTORENT | BYTE | 50 DUP(? ) |
| BOOKUSEDSTATUS | BYTE | MAX\_BOOKS DUP(0) |
| CURRENTBOOKINDEX | DWORD | 0 |
| ISNNOTFOUND | BYTE | "ISBN Not Found", 0 |
| USEDSLOTS | BYTE | MAX\_BOOKS DUP(0) |
| USERCHOICE | BYTE | 2 DUP(0) |
| OPTIONCHOICE | BYTE | 3 DUP(0) |
| INVALIDCHOICEMESSAGE | BYTE | "Invalid choice. Please enter a valid option.", 0 |
| UPDATEMSG | BYTE | "Press 1 for Updating Book Title", 0 |
| UPDATEMSG2 | BYTE | "Press 2 for Updating Author Name ", 0 |
| UPDATEMSG3 | BYTE | "Press 3 for Updating Book Quantity", 0 |
| INVALIDUPDATE | BYTE | "Invalid choice. Try again", 0 |
| CUSTOMERIDPROMPT | BYTE | "Enter Customer ID: ", 0 |
| CUSTOMERNOTFOUNDMSG | BYTE | "Customer ID not found. Please register first.", 0 |
| RENTALTITLEPROMPT | BYTE | "Enter the book title you want to rent: ", 0 |
| BOOKRENTEDMSG | BYTE | "Book has been rented successfully!", 0 |
| BOOKALREADYRENTEDMSG | BYTE | "Sorry, this book is already rented.", 0 |
| MAXCUSTOMERNAMELEN | EQU | 50 |
| MAXCUSTOMERS | EQU | 50 |
| CUSTOMERIDS | DWORD | MaxCustomers DUP(? ) |
| CUSTOMERISBNS | DWORD | MaxCustomers DUP(0) |
| CUSTOMERNAMES | BYTE | MaxCustomers \* MaxCustomerNameLen DUP(0) |
| CURRENTCUSTOMERINDEX | DWORD | 0 |
| ALOTTED | DWORD | 50 DUP(0) |
| BOOKRENTALSTATUS | BYTE | MAX\_BOOKS DUP(0) |
| BOOKRENTEDTOCUSTOMER | DWORD | MAX\_BOOKS DUP(0) |
| BOOKQUANTITYMSG | BYTE | "Available copies: ", 0 |
| NOMORECOPIESMSG | BYTE | "Sorry, no copies available for rent.", 0 |
| RENTALCOUNTMSG | BYTE | "Number of books currently rented: ", 0 |

.CODE SEGMENT

* USER-DEFINED (PROCS)

|  |  |
| --- | --- |
| MAIN | Entry point of the program; manages user input, navigates to Librarian or Customer menus. |
| ADDBOOK | Collects and stores details (title, author, ISBN, and quantity) for a new book. |
| UPDATEBOOKINFO | Displays update options (title, author, or quantity) and processes user choice to update data. |
| UPDATEAUTHORNAME | Updates the author name of a book based on the user's provided ISBN. |
| UPDATEBOOKTITLE | Updates the title of a book based on the user's provided ISBN. |
| UPDATEBOOKQUANTITY | Updates the quantity of a book based on the user's provided ISBN. |
| PURCHASEBOOK | Placeholder for a future function related to purchasing a book (not implemented). |
| RENTBOOK | Handles the process of renting a book, checks availability, and updates book status. |
| RETURNBOOK | Handles book returns, increases book quantity, and clears customer rental status. |
| DISPLAYLIBRARYINFO | Displays details of all books currently stored in the library. |
| DISPLAYBOOKINFO | Displays information about a specific book based on the ISBN provided by the user. |
| EXITPROGRAM | Ends the program execution. |
| DISPLAYWELCOMEMESSAGE | Displays a welcome message with special text effects, colors, and animations. |
| DISPLAYWELCOMEMESSAGE1 | Displays introductory welcome animations before user interaction begins. |
| INVALIDCHOICE | Displays an error message when the user makes an invalid menu choice. |
| ENDPROGRAM | Displays a final exit message and ends the program. |
| CUSTOMERLOGIN | Handles customer login by checking if the entered Customer ID exists. |
| REGISTERSTUDENT | Registers a new student with ID and name, ensuring no duplicate Student IDs exist. |

* BUILT-IN PROCS FROM IRVINE32.LIB

|  |  |
| --- | --- |
| WRITESTRING | Displays a null-terminated string on the console. |
| WRITECHAR | Writes a single character to the console. |
| WRITEINT | Writes an integer to the console. |
| READINT | Reads an integer from user input and stores it in the EAX register. |
| READSTRING | Reads a string from user input and stores it at the specified address. |
| CLRSCR | Clears the console screen. |
| CRLF | Moves the cursor to a new line in the console. |
| DELAY | Delays the execution of the program for a specified time. |
| SETTEXTCOLOR | Sets the text color for console output. |
| EXIT | Terminates the program (similar to ExitProcess in Windows). |

[UTILIZED ASM METHODOLOGIES]

* **ADVANCED PROCEDURES (STACK, ESP, EBP, LEA, LOCAL)**

1. **STACK USAGE:** Procedures like RegisterStudent and CustomerLogin utilize local storage with push and pop to save and restore registers.
2. **PROCEDURE CALLS:** Procedures use the ret instruction to return control to the caller. Procedures like DisplayWelcomeMessage1, RegisterStudent, and EndProgram use call and return logic to manage control flow.
3. **LEA USAGE:** The LEA (Load Effective Address) instruction is not explicitly seen, but effective address calculations are done using add edx, eax for array indexing.

* **CONDITIONAL PROCESSING (CMP, JMPS, LOOPS):**

1. **CONDITIONAL CHECKS:** The program relies heavily on CMP for checking user input (cmp eax, 1, cmp eax, 2) and validating array values (cmp eax, BookISBNs[esi]).
2. **JUMP INSTRUCTIONS:** It uses unconditional jumps (jmp) for control flow, like returning to MainLoop, allow, or moving to error-handling logic in UpdateBookInfo and ReturnBook.
3. **LOOP INSTRUCTIONS:** The loop instruction is used in loops like SearchLoop for scanning ISBNs in UpdateBookInfo and DisplayBookInfo.

* **STRING HANDLING (FUNCTIONS/INSTRUCTIONS SCASB, CMPSB, LODSB):**

1. **STRING COMPARISONS:** The REPE CMPSB instruction is used to compare strings in the RentBook procedure to check if a book title exists in the BookTitles array.
2. **STRING DISPLAY AND INPUT:** Strings are displayed using WriteString, and input is taken using ReadString. Examples include prompts like BookTitlePrompt, BookAuthorPrompt, and BookISBNPrompt.

* **INTEGER ARITHMETIC (ROTATE, SHIFT, MUL, IMUL) + SIGN EXTENSION INSTRUCTIONS (CBW, CWD, CDQ, IDIV, ADC, SBB):**

1. **MULTIPLICATION AND DIVISION:** Multiplication (MUL) is used extensively for calculating array indices. For example, mul ebx calculates the position of book titles in AddBook and UpdateAuthorName.
2. **DIVISION:** Integer division (DIV) is used to calculate indices when converting a byte offset to an array index, such as in the UpdateBookTitle procedure (div ebx).

* **LOOPING MECHANISMS:**

1. **LOOPS:** The program features several loops for array processing. The loop instruction is used to iterate through arrays like BookISBNs in UpdateBookInfo, ReturnBook, and DisplayBookInfo.
2. **CUSTOM LOOPS:** The program also creates loops using labels and jumps (e.g., L1: and SearchLoop:) to search for specific data, such as books, ISBNs, or customer IDs.

* **STACK OPERATIONS (PUSH, POP, PUSHFD, POPFD):**

1. **PUSH AND POP USAGE:** The program makes use of push and pop for saving and restoring the state of registers, as seen in DisplayWelcomeMessage, where eax is pushed and restored after a delay function.

* **PROC DIRECTIVES:**

1. **PROCEDURE DEFINITION:** Procedures are defined with the PROC and ENDP directives, such as main PROC and main ENDP. This structure allows for the modular design of reusable procedures.
2. **MULTIPLE PROCEDURES:** Examples include Main, AddBook, UpdateBookInfo, RegisterStudent, ReturnBook, RentBook, and CustomerLogin.

OUTPUT CONSOLE

|  |
| --- |
| 1. **WELCOME AND INTRODUCTION** |
| **Functionality:** Displays a welcome message and an ASCII art banner using the provided strings (e.g., Line1, WelcomeMessage).  **Procedure:** DisplayWelcomeMessage |
|  |
|  |
|  |
|  |

|  |
| --- |
| 1. USER LOGIN |
| Functionality:o Prompt the user to identify their role: Librarian or Customer.  Handle librarian/customer-specific options based on the choice.  Procedure: LoginPrompt  Inputs: Role selection: (1) Librarian, (2) Customer  Outputs: Success or error messages for login. |
|  |
|  |

|  |
| --- |
| 1. LIBRARIAN FEATURES |
| 3.1. Add New Book  Functionality: Add new books to the library.  Procedure: AddBook  Inputs: Title, Author, ISBN, Quantity.  Outputs:Success message or error if the library is full |
|  |
|  |

|  |
| --- |
| 3.2. UPDATE BOOK INFO |
| Functionality: Update the title, author, or quantity of a book.  Procedure: UpdateBook  Inputs: ISBN, field to update, new value.  Outputs: Success message or "Book not found" message. |
|  |

|  |
| --- |
| 3.3. DISPLAY LIBRARY INVENTORY |
| Functionality: Show all books in the library along with their details.  Procedure: DisplayLibraryInventory  Outputs: List of all books with title, author, ISBN, and quantity. |
|  |

|  |
| --- |
| 3.4. REGISTER NEW STUDENT |
| Functionality: Add a new student to the system.  Procedure: RegisterStudent  Inputs: Student ID, Name.  Outputs: Success message or duplicate ID error. |
|  |

|  |
| --- |
| 3.5 DISPLAY BOOK INFO |
| Prompts for the selected ISBN  As user inputs the ISBN, function uses compare instruction to match the ISBN to show all possible information of the book including book titile, author, ISBN, |
| A black screen with white text  Description automatically generated |

|  |
| --- |
| 3.6 CLOSE PROGRAM |
| Closes the main PROC  Jumps straight to INVOKE ExitProcess, 0.  Calls the respected function  Terminates the program code. |
| A black screen with white text  Description automatically generated |

|  |
| --- |
| 4. CUSTOMER FEATURES |
| 4.1. RENT BOOK |
| Functionality: Rent a book if available.  Procedure: RentBook  Inputs: Title or ISBN.  Outputs: Success message or error if no copies are available or already rented. |
|  |

|  |
| --- |
| **4.2. RETURN BOOK** |
| Functionality: Return a rented book.  Procedure: ReturnBook  Inputs: Title or ISBN.  Outputs: Success message or error if no book is currently rented. |
|  |

|  |
| --- |
| 4.3 LOG OUT |
| Called and seen in Customer interface, jumps back to the selected occupation interface. |
| A computer screen shot of a computer program  Description automatically generated |

[UNDERSTANDING]

Developing this library management system has deepened our understanding of assembly language programming concepts, including:

* Memory management
* Low-level data manipulation
* Input/output operations and handling
* Algorithms for searching and sorting data. This project has also reinforced problem-solving skills as challenges arose in the design and implementation of system functionalities. Through this level of debugging and testing, a comprehensive grasp of both the strengths and limitations of assembly language has been achieved.