WEB-BASED DATA BASE ASSEMBLY LANGUAGE EDITOR

A graduate project submitted in partial fulfillment of the requirements
For the degree of Master of Science in Computer Science

By

Veronika Movagharian Pour

May 2013
The graduate project of Veronica Movagharian Pour is approved:

______________________________________________  ____________
Gloria Melara, Ph.D. Date

______________________________________________  ____________
Rick Covington, Ph.D. Date

______________________________________________  ____________
Son Pham, Ph.D. Chair Date

California State University, Northridge
Dedication

This paper is dedicated to all my classmates with whom I have done group projects or presentations and who helped me learn course materials and improve my knowledge. Also to all my friends who have encouraged me to continue when this was more difficult than I thought it should have to be.
Acknowledgements

Special thanks to Professor Son Pham who helped me through this big mission. Without his unlimited help and support, it would be impossible for me to complete this project. Having the opportunity to spend hours working with him during his office hours hand in hand, I have learned lots of programming skills that not only gave me the power I needed to accomplish this hard task, but also provided me with the knowledge and the self-confidence I required to get a job as a software developer. I really appreciate all his efforts to help my dreams become true. Thank you.

I also want to thank David Brotz who agreed to be my partner for database design class project. He is a professional PHP programmer from whom I have learned lots of programming skills either. He is a very talented person who is always ready to help his friends. It is my honor to be one of his friends. Thank you so much David.

Also thanks to Professor Jeff Wiegley who provided me with three thesis / projects to use as sample. Having Read these thesis / projects, I have learned how to write and prepare my own thesis / project. Thank you.

Last but not least, thanks a lot to my family who has never let me down with love and with emotional and financial support.
# Table of Contents

Signature Page ........................................... ii  
Dedication ................................................ iii  
Acknowledgements ..................................... iv  
List of Figures .......................................... vii  
Abstract ................................................ x  

## Chapter 1  Introduction  
1.1 Assembly Editor User Types................................. 3  
1.2 Assembly Editor Main Functions............................. 5  
1.3 Summary Of Work Done For This Project...................... 9  

## Chapter 2  Working Environment And Structure  
2.1 System Functionality........................................ 12  
2.1.1 Login Process........................................ 15  
2.1.2 System Functionality Process After Login................. 16  

## Chapter 3  Available Features For DBA  
3.1 Metaclasses Management Option............................. 18  
3.1.1 Edit Button........................................ 20  
3.1.2 Delete Button....................................... 22  
3.1.3 Copy Button......................................... 23  
3.1.4 Down-Arrow Button.................................. 25  
3.1.5 New Button........................................ 25  
3.1.6 Test Code Button...................................... 27  
3.1.7 Up-Arrow Button In The Backup Table...................... 27  
3.2 Metaclasses Management Using List Link.................... 27  
3.3 Template Management Using Table Link.................... 28  
3.4 Template Management Using List Link.................... 31  

## Chapter 4  Available Features For Instructor  
4.1 Homework Generator Management Using Table Link.......... 33  
4.2 Homework Generator Management Using List Link.......... 34  
4.3 Homework Template Management Using Table Link.......... 35  
4.4 Homework Template Management Using List Link.......... 36  

v
4.5 Assembly Class Management Table Using Table Link..........................38
4.6 Assembly Class Management Table Using List Link..........................40
4.7 Student Scores Link........................................................................40
4.8 Student History Link........................................................................42
4.9 Instructor Assembly Page Link............................................................43
  4.9.1 Top Center Frame........................................................................44
  4.9.2 Center Frame...............................................................................45
  4.9.3 Bottom Frame..............................................................................48
  4.9.4 Left Center Frame........................................................................51
  4.9.5 Right Center Frame......................................................................54

Chapter 5 Available Features For Student .................................57
  5.1 Student Assembly Page.................................................................58
    5.1.1 Top Center Area........................................................................59
    5.1.2 Left Center Area.......................................................................60
    5.1.3 Center Area..............................................................................61
    5.1.4 Right Center Area......................................................................64
    5.1.5 Right Center Area......................................................................64
  5.2 Check Your Score Link....................................................................67

Chapter 6 Conclusion.........................................................................69
  6.1 Fututre Work..................................................................................69
References.........................................................................................71
List of Figures

Figure 1.1: System Data Flow Diagram.................................................................2
Figure 1.2: Code Distribution In The Entire System.............................................3
Figure 1.3: Assembly Editor User Types.............................................................4
Figure 1.4: Assembly Editor Main Operations...................................................5
Figure 1.5: Table Management Functionality....................................................6
Figure 1.6: Code Compile Functionality.............................................................7
Figure 1.7: Code Translation Functionality........................................................8
Figure 1.8: Code Edit Functionality.................................................................8
Figure 1.9: Student History/Score Functionality..............................................9
Figure 2.1: Database Schema........................................................................12
Figure 2.2: Main Program..............................................................................14
Figure 2.3: Login Process..............................................................................15
Figure 2.4: Login Page..................................................................................16
Figure 2.5: System Functionality Process After Successful Login...................17
Figure 3.1: DBA Success Message.................................................................18
Figure 3.2: DBA Control Panel Page...............................................................19
Figure 3.3: SQL Query Successful Execution Message....................................20
Figure 3.4: Metaclass Management Through Table..........................................21
Figure 3.5: Metaclass Management Through Table..........................................21
Figure 3.6: Metaclass Backup Management Through Table............................22
Figure 3.7: Metaclass Edit Page.........................................................................23
Figure 3.8: Can’t Delete Message......................................................................24
Figure 3.9: Metaclass Delete Page....................................................................24
Figure 3.10: Successful Delete Message.........................................................25
Figure 3.11: Metaclass New Page......................................................................26
Figure 3.12: Successful Insertion Message.......................................................26
Figure 3.13: Metaclass Test Code Page.............................................................27
Figure 3.14: DBA Metaclass Interface Page.....................................................28
Figure 3.15: Template Interface Page...............................................................29
Figure 3.16: Template Edit Page.......................................................................29
Figure 3.17: Template Delete Page....................................................................30
Figure 3.18: Template New Page....................................................................30
Figure 3.19: Template Test Code Page.............................................................31
Figure 3.20: Template Interface Page ................................................................. 32
Figure 4.1: Instructor Success Message .............................................................. 33
Figure 4.2: Instructor Control Panel Page ........................................................... 34
Figure 4.3: Homework Generator Interface Page .................................................. 34
Figure 4.4: Homework Generator Interface Page .................................................. 35
Figure 4.5: Homework Generator Interface Page .................................................. 36
Figure 4.6: Homework Template Interface Page .................................................. 37
Figure 4.7: Homework Template Interface Page .................................................. 37
Figure 4.8: Homework Template Interface Page .................................................. 38
Figure 4.9: Instructor Assembly Classes Interface Page ......................................... 39
Figure 4.10: Instructor Assembly Classes Interface Page ....................................... 39
Figure 4.11: Instructor Assembly Classes Interface Page ....................................... 40
Figure 4.12: View Student Scores Page ............................................................... 41
Figure 4.13: View Student Scores Page For A Selected Assignment ...................... 41
Figure 4.14: View History Page ........................................................................... 42
Figure 4.15: View History Page For A Selected Student ....................................... 43
Figure 4.16: Instructor Assembly Page (Assembler Editor Page) ............................ 43
Figure 4.17: Student Solution Displayed In The Left Textarea On The Center Area .... 44
Figure 4.18: Message Informing There Is No Solution Submitted .......................... 45
Figure 4.19: Hex Code In The Right Textarea ....................................................... 46
Figure 4.20: Octal Code In The Bottom Textarea .................................................. 46
Figure 4.21: Binary Code In The Bottom Textarea ............................................... 47
Figure 4.22: Loaded Ram With Hex Code ............................................................. 47
Figure 4.23: CPU Registers In Cycle 1 Using RunToEnd Button ............................ 48
Figure 4.24: RAM After Run Using RunToEnd Button ......................................... 49
Figure 4.25: CPU Registers In Cycle 10 Using RunToEnd Button ......................... 49
Figure 4.26: CPU Registers In Cycle 1 Using Run Button ..................................... 50
Figure 4.27: CPU Registers In Cycle 1 Using Run+Ram Button ............................ 51
Figure 4.28: Assembly Command Edit Page ....................................................... 52
Figure 4.29: Opcode Duplication Warning Error .................................................. 52
Figure 4.30: Assembly Command Delete Page .................................................... 53
Figure 4.31: Assembly Command Create New Page ............................................ 53
Figure 4.32: Login / Password Edit Page ............................................................. 54
Figure 4.33: Login / Password Delete Page ......................................................... 55
Figure 4.34: Login / Password Create New Page ................................................ 56
Figure 5.1: Student Success Message..............................................................57
Figure 5.2: Student Page ..................................................................................58
Figure 5.3: Student Assembly Page (Assembler Editor Page)..............................59
Figure 5.4: Student Solution In The Left Textarea In The Center Frame..............60
Figure 5.5: Information Table Of Add Command In The Bottom Frame.............61
Figure 5.6: Hex Code In The Right Textarea.......................................................62
Figure 5.7: LOADED RAM With The Hex Code Loaded.................................62
Figure 5.8: Octal Code In The Bottom Textarea..................................................63
Figure 5.9: Binary Code In The Bottom Textarea..............................................63
Figure 5.10: CPU Registers’ Contents In Cycle 1 Using RunToEnd button.........64
Figure 5.11: Ram Content After Execution Using ViewRamAfterRun...............65
Figure 5.12: CPU Registers’ Contents In Cycle 10 Using RunToEnd button........65
Figure 5.13: CPU Registers’ Contents In Cycle 10 Using Run button................66
Figure 5.14: CPU Registers’ Contents In Cycle 10 Using Run+Ram button..........67
Figure 5.15: Scores Table Of Student John Smith..........................................68
ABSTRACT

WEB-BASED DATA BASE ASSEMBLY LANGUAGE EDITOR

By

Veronika Movagharian Pour

Master of Science in Computer Science

Assembly programming language is taught to the students with computer science major in COMP 122 Computer Architecture course. Since assembly programming languages are considered low level programming languages, it is pretty challenging to learn and use Assembly languages in programming. In an effort to better understand and use assembly programming languages, this project utilizes a web-based database tool, Database Assembly Editor that enables the visualization of the result of executing each assembly instruction in the CPU and RAM. This software is further developed to extend its functionality in assisting both students and instructors in managing the course assignments (submitting and grading) via web through PHP application.

The thesis starts with introducing this software system demonstrating how it works followed by explaining basic features and structures used to create this Assembly Editor software. In the proceeding chapters, the available features to use by each user type are introduced in detail along with screenshots illustrating each discussed scenario. Having read this documentation and been provided with a login and password, a user can simply access this software through browser and use the tool features available to him based on his predefined user type.
Chapter 1

Introduction

My Project is to create a computer Editor to allow users to write, compile and run Assembly Language Programs. The Project is also to allow the Assembly Programs to be stored in the database with the facility of program reload for editing and grading purposes.

My Assembly Editor is web-based meaning that users can access it anytime from anywhere through Internet connection with an authentication system. This system enables many users to access and use the Editor at the same time, each with their own account, without any collusion.

My Assembly Editor is designed on a 3-tier system architecture: Web Server, Data Base Management System (DBMS), and Web Browser provided by Web Services. For this project, I have specifically used PHP as Web Programming Language for the Web Server, MySQL as my DBMS along with Operating System Linux with Apache.

The way the 3-tier system works is that the user connects to the Web Server through a Web Browser using his local PC requesting for the Assembly Editor Service. The Server connects to the DBMS to authenticate the user. If the Authentication process succeeds, meaning that the DBMS confirms the user, the Server allows the user to access and use the Assembly Editor. To access and use my Editor, user can use any Browser such as Firefox, Internet Explorer, and Chrome with no limitation.

The architecture I have used to design and develop this Assembly Editor is to put the whole source code in the database along with a short index.php file located in the Server that runs the source code based on the user request each time. In each step, the index.php file on the Server checks the user request, connects to the database, fetches proper source code, runs it and displays the associated Graphical User Interface (GUI) to the user. Figure 1.1 displays the data flow between the user and the Assembly Editor system.

Though PHP is not considered an Object Oriented Programming Language like Java, I have strictly used Object Oriented method to create my Assembly Editor simply by defining classes along with the necessary functions for each class, and then, using these classes in the source code for GUIs.

In the database, I have created two main tables that interact with the index.php file to manage the whole system execution. One of these tables stores the classes and the other one keeps the source code for each web page that should be run and displayed to the user based on his request. Basically, each user request triggers the index.php file with a key that identifies the web page that should be displayed to the user. The index.php file then, connects to the GUI table and performs a search for the web page with the identification that matches the triggered key. After finding the proper web page, index.php file fetches the associated source code and runs it leading to displaying user requested web page. Figure 1.2 displays the code distribution along the entire system.
As mentioned before, all classes that are being used in the source code are stored in the class table. I have used two types of classes defined in the database class table for my project: There are classes that assist the index.php file to pull out and execute the source code to display proper web pages to the user, and there are classes that are being used to compile and run Assembly Language Programs by defining the actions that should take place when running Assembly Programs.

In this project, I have designed all classes along with the necessary functions required to authenticate users, find and fetch the PHP code for the proper GUI (web page) based on the user request from the database, execute the code, perform required actions and finally display the requested GUI along with a confirmation message if needed. I have also designed the database and written the code behind each webpage. However, the classes that determine the necessary actions to take place when running Assembly Programs are received from Professor Pham and I have just used them in my code when needed.
1.1 Assembly Editor User Types

My Assembly Editor has basically three types of users: Student, Instructor, and Data Base Administrator (DBA). In general, when the user connects to the server requesting using the Assembly Editor, the Server, not only checks for the user authentication through the index.php file, it also determines the user role or type in order to provide the user with the proper available features designed for that specific user type. Figure 1.3 displays the user types in my Assembly Editor System.

A user with DBA role is empowered to have access to the both Class and GUI tables allowed to modify the source code through Table Management designed for DBA. A DBA user is the person who has the responsibility of providing user friendly interface. The purpose of the DBA Table Management System is to allow database table contents’ modification through DBA user interface. I have designed this system to enable the DBA manipulate the database tables’ contents through the Editor without the necessity of performing the manipulations by directly accessing the database. The features designed in this system will be explained in more detail in chapter 2 and chapter 3.
A user with Student role is empowered to access my Assembly Editor to write, compile, and run Assembly Language Programs. He is able to pull out homework assignments prepared by the instructor and submit the solution program for each assignment. He can also pull out and edit his previously submitted solutions if not yet graded by the instructor. The student is also provided with programming assist displaying right format for Assembly commands and some useful information. The student has also the feature that displays the grades earned for homework assignments along with the solutions submitted. All the mentioned features are available to the students through GUIs I have designed for this purpose. More detail regarding GUIs and features designed for a student user will be in chapter 2 and chapter 4.

A user with Instructor role has the features that allow him to create, save, and activate homework assignments on line using his local PC. The instructor is able to pull out students’ solutions submitted for each homework assignment and grade them. The grade to each solution for each student will be saved next to that solution in the student account independent to the other students. The instructor is also provided with the features that allow him to compile and run Assembly Programs the same way a student does with the difference that the instructor usually does not write the programs. Instead, he pulls out students’ programs from the database to compile and run in order to grade them. However, he is also able to compile and run his own written Assembly programs if he
wants to. In chapter 2 and chapter 5, these GUIs and features will be explained in more detail.

1.2 Assembly Editor Main Functions

All the operations that can be performed through the Assembly Editor can be summarized into five main functions including table management, code compile, code translation, code edit, and student history/score display. Figure 1.4 displays Assembly Editor main functions.

![Figure 1.4: Assembly Editor Main Operations](image)

Table Management includes edit, delete, create new, and copy operations. It allows the user to perform Read and Write operations on a database table. All user types have table management feature but on different tables including all or part of the available operations.

When the user requests for a table management operation through the available buttons or hyperlinks, the server finds the proper table and performs the requested operation. Then,
the server displays the proper GUI along with a confirmation message confirming that the operation is performed successfully. Figure 1.5 displays table management functionality.

![Table Management Diagram](image)

**Figure 1.5: Table Management Functionality**

Code compile facility is provided for a user with student or instructor role. The instructor usually fetches the students’ code from the solution database table to compile, though he can also compile his own code. But when the student wants to compile a code, he should either write the code into the provided textbox or should reload his previously submitted code from the solution database table. The code should be written in the right format as defined in the system. To write the code in the right format, the student can use the instructionset list available to him.

When the user requests for code compile, the server fetches required classes and functions from class table and then, compiles the code to hex code displaying the proper GUI containing the result to the user. Figure 1.6 displays Code Compile functionality.

Code Translation feature is also available to a user with student or instructor role. Basically, when the user requests for code translation to octal or binary code, the server, fetches required classes and functions from class table and translate the code to its octal
or binary equivalent code. The server then, displays the result through the GUI designed for this purpose. Figure 1.7 displays code translation functionality.

**Figure 1.6: Code Compile Functionality**

When a student or the instructor requests for code edit, the server fetches classes and functions needed for editing the code, takes the hex code and the input if any, performs the edit operation using these classes and functions, and displays the CPU, RAM and output values. Figure 1.8 displays the code edit functionality.

Finally, when requesting for student scores by the student or requesting for students’ history/score by the instructor, the server performs a search on the Solution table and fetches requested information from the table, displaying the result through a GUI specified using the code behind the web page that allows the user to send the request. Figure 1.9 displays the student history/score functionality.
Figure 1.7: Code Translation Functionality

Figure 1.8: Code Edit Functionality
Each time the user sends a request through a button or a hyperlink, the code behind it determines what is required to be fetched from the database, what operation is needed to be performed and which GUI along with what result should be displayed to the user.

In each step, if the user requests to write to the database, for example, the student requests to save his solution or the instructor requests to save the grade for a student solution to a homework assignment, the system will return a confirmation message ensuring the user that the operation is performed successfully. When the user is ready to sign out from the Assembly Editor System, he can simply select Log Out link available to him. This way, he can securely leave the system.

My Assembly Editor is an easy to use software available to students to assist them learn and practice Assembly language programming along with a simple and convenient way to submit their solutions to the homework assignments assigned to them by the instructor. The following chapters will provide the readers with more detail about this system.

1.3 Summary Of Work Done For This Project

The software package of the thesis is Assembly Editor, a web based software with a support of database and Object Oriented Programming method in PHP. The functionality
of this software is based on a framework which is a collection of classes stored in a database table (Metaclass table), a database table that keeps code behind GUls/web pages (Template table), and a main PHP file, index.php located on the server. The interaction between index.php file and Template table using classes from Metaclass table enables the framework to run this software package. Following is a short description of what I have done in this project to create my Assembly Editor:

1. In the first step to create my Assembly Editor, I have implemented the framework to enable my software functionality. This framework empowers the software to find the proper code from the database each time a web page is requested, attach required classes (if any) to the code and execute the code to produce the desired results. To accomplish this, I have performed following steps:

   - Created ‘Metaclass’ table, a database table, to save classes and their functions. Each class is accessible to the code through querying this table using the class name. For each class, a record is created in this table saving the class name in one field and the class body (variables and functions) in another field, allocating it a unique number to be used for class identification.

   - Created ‘Template’ table, a database table, to store code behind each GUI/Web page. When a web page is requested by the user, the code behind that web page is accessible by querying this table. For each web page, a record is created that keeps the code behind the web page in one field and the name of the classes needed for running the code in the ‘requiredclasses’ field along with a key to be used to access the web page.

   - Created ‘AnalyzeCode’ class to find and save class names that should be attached to each web page before execution. Each time changes are applied to a web page code, this class gets called. The class is used to read the code behind the web page to distinguish all the class names in the code and record these class names in the ‘requiredclasses’ field.

   - Created ‘AnalyzeClass’ class to check class names in the ‘requiredclasses’ field in the Template table. When the code behind a web page gets modified, this class is used to check ‘requiredclasses’ field. If the user enters the required class names manually and mistypes a class name, this class will detect and notify the user preventing this class name to be stored.

   - Created ‘Template’ class to find the proper web page from Template table, load required classes from Metaclass table and run the code. This class is gets called by the main php file (index.php) and assists in finding and displaying web pages.

   - Created index.php file to find and execute proper code based on the user request, and to display the associated web page using Template class. This PHP file calls Template class and uses its functions to load the requested web page.
- Created ‘User’ class that helps the Assembly Editor software to authenticate the user. This class takes the user’s supplied login and password determining whether the user is an authorized user.

2. In the next step, the software requirements are determined and the overall software layout including database tables is designed. This is the time when specified what operations this software is going to perform and which operations should be available to each user type. The GUIs/web pages are also designed in this stage of the software development process.

3. Then, I have received classes required for code compile, translate, edit and run from Professor Pham and saved them in the Mataclass table making them available in the project. These classes are used to perform compile, translate, edit, and run Assembly code operations only and are not included the code to display the operation results.

4. And, finally the code behind all GUIs along with their features are written based on my design using classes and their functions from Metaclass table when necessary. The code for displaying each GUI/web page is written in PHP having HTML and Java Script code embedded. Also, to query the database tables and fetch required information SQL (Structured Query Language) is used.
Chapter 2

Working Environment and Structure

My Assembly Editor is implemented in a 3-tier system architecture: Web Server, Database Management System (DBMS), and Web Browser. For this project, I have specifically used PHP as Web Programming Language for the Web Server, MySQL as my DBMS along with Operating System Linux with Apache. To display the GUIs / web pages, HTML and Java Script code are embedded into PHP code.

The main PHP file, index.php, that runs the entire system, is located on the Web Server. However, the source code that is used to display web pages is stored in the Template table in the database. Thus, there is a kind of coupling between index.php file and Template table. The functionality of my Assembly Editor is entirely based on interaction between these two objects.

My Assembly Editor also uses some classes stored in the Class table named Metaclass to execute and display web pages. I have designed these classes including some functions to assist the index.php file to fetch and execute source code from the Template table and display web pages. There are also classes that enable the Editor system to compile, translate and run assembly code. These classes are saved in the Metaclass table as well.

To better understand the Assembly Editor system structure, let’s take a look at the database tables and their design. Figure 2.1 displays the database schema.

Figure 2.1: Database Schema
As discussed before, Template table stores the source code for web pages displaying through the Assembly Editor system. The combination of class and function fields is used as the unique key that helps to identify the requested web page. When a web page is requested by the user, the requested web page key is set and then compared to the web page keys list in the Template table. When a match is found, the code saved in the processing field is fetched and executed resulting in displaying the requested web page.

Metaclass table keeps the predefined classes and functions assigning them a unique id that is used to access and use by the index.php to perform certain operation in the entire system.

User table contains users’ information including their login and password to be used for authentication and role that determines which options and features also which web pages to present to the user.

Homework table keeps the homework assignments designed by the instructor including active field which is used to prohibit displaying inactive homework assignments to the students. This option provides the instructor with the possibility of creating assignments any time with the option of displaying them to the students later. The value of this field is either 0 meaning inactive or 1 meaning active.

Hw-Generator table is designed to allow the instructor create a specific GUI for a specific homework assignment. For example, if a homework assignment requires many inputs, a GUI should be displayed to the student including some textboxes that allow him to enter the inputs. In this case, the instructor can create the GUI and save it in this table associating to that specific homework assignment. Though I have designed this table, I did not use it in this project. Therefore, there will not be more detail about this table. It is good to know however, that this feature is already designed and ready to use.

Solution table holds the solution and the grade to homework assignments. When saving solution to an assignment by a student, the student_id (student unique identification id) and hw_id (homework assignment unique id) which are already known by the system are also saved next to the solution. The combination of student_id and hw_id is then being used as key to find and fetch the solution and grade if it is already graded.

Instructionset table contains predefined instructions/commands that are recognizable in the system. All the commands listed in this table can be used in programs. It is possible to extend the list and add to the existing commands in this table. However, the added commands should be defined in the CPU class that is being used to compile and run Assembly code. This table is to assist students with formatting commands and using them in coding.

The main program that runs the tool is only one page long. It uses the two variables class and function, and by evaluating their value and comparing them to the class and function fields in the Template table determines which page should be displayed. Figure 2.2 displays the database schema.

The rest of the program is organized in two tables: one template table for GUls named Template Table and one class table containing classes used in the entire system named Metaclass Table. The Template Table has two fields: class and function that are being
used by the main program to find the related GUI in each case. Necessary classes are loaded during the execution of each subprogram if needed. Remaining tables are to save and retrieve necessary information throughout the system.

**Figure 2.2: Main Program**

The designed database is an object oriented database and the CPU, RAM, Input, Output, and Instructions are all treated as object. There are classes for CPU and RAM along with necessary procedures that perform the expected behavior when running Assembly code. There are also other classes that are being used through the instructions’ execution.
2.1 System functionality

This assembly language editor is a web-based application that does not require having any specific programming language installed on the computer to execute the program. However, it requires connecting to the CSUN Virtual Private Network and having SSH secure shell. Every person who has a login and password assigned can run the program using link: http://jd-research.ecs.csun.edu/~emulator/Ass_index.php through a browser like Firefox. This link then will direct the user to the Login page and after entering the assigned login and password, the user will be presented to the appropriate page based on the determined user type.

2.1.1 Login Process

The communication between the user and the Assembly Editor takes place through browser. However, to simplify the system and to better understand how the system works, I have eliminated the browser from the system. Just remember that the browser plays the communicator role during the whole system operation. Figure 2.3 displays the login process.
Here are the steps taken to login into the Assembly Editor software and start to use it:

1. The user browses the link in the browser leading the index.php to run.
2. The Server fetches Template class from Metaclass table and uses it to fetch web pages.
3. The Server fetches the Login page.
4. The Server displays the Login page to the user.
5. The user enters his login and password.
6. The Server authenticates the user and set his Role.
7. The Server fetches the default web page for the user based on his Role.
8. The Server displays the web page to the user.

Figure 2.4 displays the login page. In the proceeding chapters, the available features for each user type are explained in detail and the associated figures are displayed.

![Login Page](image)

**Figure 2.4: Login Page**

### 2.1.2 System Functionality Process After Login

After successful login, the user is provided with the web page containing available features based on his role. The available features are presented through links and buttons. The user can send his requests to the system by using these links and buttons. Each time the user selects a link or a button, the index.php file evaluates his request using Template class loaded before. If any web pages are needs to be displayed, the index.php file fetches
the web page from Template table using the key triggered when sending the request and displays it to the user. If any classes are required for the code behind the web page to be fetched and executed, they will be fetched from the Metaclass table and attached to the web page code before code execution. The name of the required classes is available in the ‘requiredclasses’ field for each web page in the Template table. If any other information, like homework description is required, it will be read from the determined table through querying that table. If any information is added to or changed in the database, or in other word, write/edit operation is performed on any database tables, a confirmation message will be displayed to the user saying that the operation is performed successfully. Figure 2.5 displays the system functionality process after successful login.

Figure 2.5: System Functionality Process After Successful Login
Chapter 3

Available Features for DBA

Upon successfully login as DBA user, the Success message will appear followed by DBA page link. This link provides access to the DBA Control Panel Page (Figure 3.1).

Figure 3.1: DBA Success Message

After clicking DBA page link, DBA Control Panel Page displays followed by all the available options for a DBA user including Metaclasses Management, Template Management, log out, and SQL Query Form options. Figure 3.2 displays this page.
The textarea of SQL Query Form allows the DBA to test SQL queries by simply typing the query in the textarea and clicking on Submit button. If the query is runnable, a message will pop up saying Successful Execution. Figure 3.3 displays this message. This is a way of allowing the DBA user to check the query of interest for operations.
3.1 Metaclass Management Option

This option is used by DBA user to manage and maintain Metaclass table. There are two choices of accessing Metaclass table within this option: one is through the Table link which allows accessing Metaclass table records through table directly. Or, List link for allowing the DBA user to choose from list of available records in the Metaclass table. The DBA has the capabilities to edit, delete, copy, backup, create new, and test code options as part of maintaining and managing Metaclass table. Figure 3.4 and Figure 3.5 display one of the Metaclass table records along with Edit, Copy, and Delete GUI buttons.
Figure 3.4: Metaclass Management Through Table

Figure 3.5: Metaclass Management Through Table
There is also a Metaclass Backup table which keeps all the Metaclass table records that are inactive and are not a parent record of other record(s). Figure 3.6 displays one record from the Metaclass Backup table.

![Image of Metaclass Backup Table]

**Figure 3.6: Metaclass Backup Management Through Table**

Records that are in the Metaclass Backup table cannot be edited or deleted in this table, but, can easily return back to the Metaclass Main table using up-arrow GUI button.

**3.1.1 Edit Button**

When clicking on the Edit button of a record in the Metaclass Main table, a page gets opened displaying the record fields allowing the user to edit body, Required Metaclasses and Active fields of that record. After completing the edit task, the user can save the result by clicking on the Submit button. However, going back using Go Back link will ignore the changes without saving the work. Figure 3.7 displays this page.
3.1.2 Delete Button

If a record in the Metaclass Table is an active record, (the Active field has value true), and we try to delete it by clicking on the Delete button, a message will pop up saying that this record is an active record and therefore, you cannot delete it. Figure 3.8 displays this page.

However, if we try to delete an inactive record in the Metaclass Table by clicking the Delete button, a page will open displaying the record fields and Confirm Deletion button. Figure 3.9 displays this page. Clicking Confirm Deletion button will result in deleting the selected record. After deleting a record, a confirmation message on the screen will indicate the success of the delete operation. Figure 3.10 displays this message.

Clicking on the Main Metaclasses Page will return the user to the DBA Metaclasses Interface Page displaying the Metaclass Main Table. However, the Go Back link will return the user to the Metaclass Main table ignoring delete operation.
Figure 3.8: Can’t Delete Message

Figure 3.9: Metaclass Delete Page
3.1.3 Copy Button

The Copy GUI button allows the user to make a copy of a record. This is useful for code reuse. Copying a record for reusing code in this case, can prove to be more efficient specially when editing is to a minimal. Thus, the new record can be quickly created by applying the necessary changes without requiring typing the same code. This can save development and troubleshooting time tremendously.

3.1.4 Down-Arrow Button

This button allows the user to remove a record from the Metaclasse Main Table and save it in the Metaclasse Backup Table. This is true only when the selected record is an inactive record and is not a parent record of other records in that table.

3.1.5 New Button

This button allows the DBA user to create new records in the Metaclasse Main Table. By clicking the New button, Metaclasses Create New Page will be opened with a textarea allowing the user to type the contents of the new record. The user can also determine the required Metaclasses for the new class manually. Then, the user can use save button to store the newly created record in the Metaclasse table, or go to the previous page using the Go Back link. Figure 3.11 displays this page.
If the user saves the new record using the Save button without determining required Metaclasses, the required classes will be determined automatically and added to the required Metaclasses field. When the record insertion process is done, a message will appear confirming success. Figure 3.12 shows this message. Main Homework Generator Page link will direct the user back to the DBA Metaclasses Interface Page.

**Figure 3.11: Metaclass New Page**

**Figure 3.12: Successful Insertion Message**
3.1.6 Test Code Button

This button allows the DBA user to run and debug the developed code. Clicking this button will take the user to the Metaclasses Test Code Page window. This window has a textarea for code entry, and Run and Save buttons. Run button will run the code within the textarea to determine required Metaclasses types. The user then can decide to save this code in the Metaclass table using save button. Figure 1.13 displays the Metaclasses Test Code Page. The Go Back link navigates the user back to the DBA Metaclasses Interface Page.

![Figure 3.13: Metaclass Test Code Page](image)

3.1.7 Up-Arrow Button In The Backup Table

This button allows the user to remove a record from the Metaclass Backup Table and save it back in the Metaclass Main Table.

3.2 Metaclasses Management Using List Link

Metaclasses Management List link assists the DBA user navigate to the DBA Metaclasses Interface Page having Metaclass Main Table List available in a dropdown list to select from. The available buttons to use are: Edit, Delete, Copy, Backup, New, and Test Code. The operations of these GUI buttons are identical to buttons used in the
Metaclasses Management through Table. This Page has also a dropdown list for Metaclasses Backup Table records. Figure 3.14 displays this page.

![DBA Metaclasse Interface Page](image)

**Figure 3.14: DBA Metaclasse Interface Page**

Backup button is equivalent to down-arrow button and Restore button is equivalent to the up-arrow button in the Metaclasse management using Table list.

### 3.3 Template Management Using Table Link

This link directs the DBA user to the Template Interface Page displaying Template table with all its records. Figure 2.15 illustrates this page. This page provides the DBA user with Edit, Delete, New, and Test Code buttons performing the same functions as the ones in the DBA Metaclasses Interface Page. The Edit button allows editing existing records in the Template table. Delete button deletes the selected record from the Template table after confirmation. The New button is used to create a new record in the table referenced above, and the Test Code Button does the same code testing operation described in the previous section as well. Figures 2.16 to 2.19 display pages that demonstrate these buttons. Also here after each successful edit, delete, create new, and test code operation, an appropriate message pops up showing the success of the performed operation. If for any reason the process does not get completed, a message will inform the user that the process failed.
Figure 3.15: Template Interface Page

Figure 3.16: Template Edit Page
Figure 3.17: Template Delete Page

Figure 3.18: Template New Page
3.4 Template Management Using List Link

This link directs the DBA user to the Template Interface Page displaying a dropdown list containing records of the Template table. The user can select a record he wants to edit or delete and then click on the appropriate button determined with a name conveying its operation. Edit and Delete buttons perform the same operations explained in the previous sections. The same thing is also valid for the New and the Test Code buttons. Figure 3.20 displays the Template Interface Page with the dropdown list of Template table records. The Main DBA Page link directs the user back to the DBA Control Panel Page.
Figure 3.20: Template Interface Page
Chapter 4

Available Features for Instructor

If a user with role Instructor successfully logs in, a success message will appear along with the Instructor page link that will direct the user to the Instructor Control Panel Page. Figure 4.1 displays this page.

By clicking on the Instructor page link, the user will be directed to the Instructor Control Panel Page where he can select between features available for the role Instructor. Figure 4.2 displays this page.

A user with Instructor role is able to manage Homework Generator, Homework Template, and Assembly Classes Tables using related Table or List link. He can also use the Student Scores link which displays the student scores for a selected assignment or the Student History link which shows the scores earned by a selected student in all assigned assignments. In the following pages, the available features for a user with instructor role are described in more detail.
4.1 Homework Generator Management Using Table Link
Figure 4.3 illustrates the Homework Generator Interface Page. The Instructor is provided with Edit, Delete, Copy, down-arrow, New and Test Code buttons with the same functionalities explained in the previous chapter. We have also Homework Generator Backup Table that keeps inactive homework items meaning that the Instructor has the ability of creating homework assignments making them invisible to students until he decides to activate these homework assignments. Inactive homework assignments can also stay in the main table instead of in the backup table with the active field set to false. Figure 4.4 displays the New and Test Code buttons along with the Homework Generator Backup Table that is currently empty. The Go Back link directs the user back to the Instructor Control Panel Page.

4.2 Homework Generator Management Using List Link

Figure 4.5 illustrates the Homework Generator Interface Page having a dropdown list for the records of Homework Generator Table. This page provides the Instructor with the ability to select a record from the list and then perform desired operation using the appropriate button. The buttons available in this page are: Edit, Delete, Copy, Backup, New, and Test Code. The function of each button is similar to the equivalent button explained in the previous chapter. This page also contains the list of records in the Homework Generator Backup Table if there is any. The Main Instructor Page link directs the Instructor back to the Instructor Control Panel Page.
4.3 Homework Template Management Using Table Link

This table contains the homework assignments that should be done using Assembly Editor. This table is designed specifically for homework assignments requiring coding in assembly language. Like the Template Table introduced in the previous chapter, this table comes with edit, delete, creating new records, Test Code buttons with the same functionality explained for each button before. Figure 4.6 and Figure 4.7 display the Homework Template Interface Page containing the records of the Homework Template Table along with Edit, Delete, New, and Test Code buttons. The Go Back link returns the user to the Instructor Control Panel Page.
Figure 4.6: Homework Template Interface Page

Figure 4.7: Homework Template Interface Page
4.4 Homework Template Management Using List Link

Figure 4.8 shown below displays the Homework Template Interface Page containing the Homework Template Table List. As it is obvious from Figure 4.8, Edit, Delete, New, and Test Code buttons are available to the Instructor use when having an item selected from the Homework Template list.

![Homework Template Interface Page](image)

Figure 4.8: Homework Template Interface Page

4.5 Assembly Class Management Table Using Table Link

This link allows the Instructor to perform limited operations on the assembly related classes kept in the Metaclass table. Though these classes are also in the Metaclass table, the Instructor cannot access all records saved in this table but only the ones that are directly related to the assembly language related operations. There is a field named access which is equal to instructor if it should be accessible to the instructor. Otherwise, this field is filled with DBA meaning that the class is only allowed to be accessed and manipulated by a user with role DBA. Figure 4.9 and Figure 4.10 display the Instructor Assembly Classes Interface Page containing a table of Metaclass table records accessible by the instructor along with DoEdit, DoDelete, Copy, Down-Arrow, Create New, Test Code buttons, and the Backup Table with a Down-Arrow button. All buttons have the same functionality as their equivalent buttons explained in the previous chapter. The Main Instructor Page link directs the user back to the Instructor Control Panel Page.
Figure 4.9: Instructor Assembly Classes Interface Page

Figure 4.10: Instructor Assembly Classes Interface Page
4.6 Assembly Class Management Table Using List Link

Figure 4.11 illustrates the Instructor Assembly Classes Interface Page containing a dropdown list of the Assembly classes with DoEdit, DoDelete, DoCopy, DoBackup, Create New, and Test Code buttons with functionalities similar to their equivalent buttons explained in the previous chapter, and the Backup Table List with the Restore button.

![Instructor Assembly Classes Interface Page](image)

**Figure 4.11: Instructor Assembly Classes Interface Page**

4.7 Student Scores Link

This link directs the user with the Instructor role to the View Student Scores Page containing a list of available active homework assignments to select from for displaying the students’ scores for the selected assignment and the View Scores button that displays the students’ scores. Figure 4.12 displays this View Student Scores Page before selecting any assignment to display the scores and Figure 4.13 shows the students’ scores earned for the selected assignment. At the top of the result table the name of the selected assignment is illustrated. The Main Instructor Page link returns the Instructor to the Instructor Control Panel Page.
Figure 4.12: View Student Scores Page

Figure 4.13: View Student Scores Page For A Selected Assignment
4.8 Student History Link

This link directs the user with the Instructor role to the Student History Page allowing the Instructor to select a student from the list and display his/her scores for all visible and active assignments by clicking View History button. Figure 4.14 displays the Student History Page before selecting a student and displaying his/her scores and Figure 4.15 displays the result table after selecting a student from the list and hitting the View History button. At the top of the result table, the name of the selected student will also be displayed.

If a student has not submitted any solution and therefore has no score earned, then the score field will display 0. Also it is possible that the submitted solution is wrong and earned therefore score 0.

![Student History Page](image)

Figure 4.14: View History Page
Figure 4.15: View History Page For A Selected Student

4.9 Instructor Assembly Page Link
Figure 4.16 illustrates the Instructor Assembly Page (Assembler Editor Page). Using the facilities provided in this page, the instructor can extract students’ solutions to different homework assignments, run the solution code one by one and give grades. In the following you can find detailed explanation of what exactly can be done in this page.

4.9.1 Top Center Frame

In this frame, the Instructor is provided with a dropdown list of active assignments and a dropdown list of all students’ names to select from. After selecting an assignment and a student name from the related lists, Display Solution button will extract the selected student’s solution to the selected assignment and displays it in the left textarea on the center area designed for this purpose. If the assignment has already scored by the Instructor, the assigned score will be displayed in the textbox next to the Score button. Otherwise, this textbox will remain empty until the Instructor attempts to score the solution. Figure 4.17 illustrates a student solution to an assignment displayed in the left textarea on the center area. The assigned score is visible in the textbox next to the Score button. If the selected student has not submitted any solution to the selected homework assignment, a message will pop up saying: This student has not submitted any solution for this assignment yet. Figure 4.18 displays this message. The Go Back link directs the Instructor back to the Instructor Control Panel Page.

Figure 4.17: Student Solution Displayed In The Left Textarea On The Center Area
4.9.2 Center Frame

As explained in the previous section, the left textarea displays solution submitted to the selected homework assignment by the selected student. The Assemble button then can be used to convert the code written in assembly language in the left textarea to its equivalent in Hex code. The result (Hex Code) will be shown in the right textarea. There are classes in the Metaclass table that are used to convert each assembly command to its equivalent in Hex. Figure 4.19 displays the Hex equivalent in the right textarea for a sample code.

Button Oct can be used to display the equivalent code in octal system in the textarea located in the bottom of the center area. Figure 4.20 displays the octal code in the mentioned textarea.

Button Bin can be used to display the equivalent code in binary system in the textarea located in the bottom of the center area. Figure 4.21 displays the binary code in the mentioned textarea.

After converting the code in the left textarea to its Hex equivalent and displaying it in the right textarea, the Load button can be used to load the Hex code into the RAM which will be displayed in the LOADED RAM textarea on the bottom area. Figure 4.22 displays the loaded RAM textarea with the Hex code loaded.
Figure 4.19: Hex Code In The Right Textarea

Figure 4.20: Octal Code In The Bottom Textarea
Figure 4.21: Binary Code In The Bottom Textarea

Figure 4.22: Loaded Ram With Hex Code
4.9.3 Bottom Frame

Loaded RAM textarea keeps the Hex code that is going to be executed. The Input textbox on the right is available to the Instructor to set the input if necessary. The output textbox on the right displays the code execution output if there is any. Between input and output textboxes there are three buttons for running the Hex code located in the Loaded RAM as follow: RunToEnd, Run, and Run+RAM.

Button RunToEnd allows the Instructor to run the Hex code and see the CPU Registers’ contents in each cycle by clicking the button with associated number. Figure 4.23 displays the CPU Registers’ contents in cycle 1.

The ViewRamAfterRun button displays the content of the RAM after code complete execution. Figure 4.24 displays the Ram content after code execution.

The CPU with its ten registers displayed at the bottom of this area contains the final values of each register in the associated textboxes. If you compare the content of these registers with the value of the equivalent registers presented in the last cycle, they should match. Figure 4.25 displays the CPU registers’ contents of the last cycle to compare with the equivalent registers displayed at the bottom of this area.

![Figure 4.23: CPU Registers In Cycle 1 Using RunToEnd Button](image-url)
Figure 4.24: RAM After Run Using RunToEnd Button

Figure 4.25: CPU Registers In Cycle 10 Using RunToEnd Button
Button Run allows the Instructor to run the Hex code and see the CPU Registers’ contents in each cycle before and after fetch, and after execution by clicking the button with associated number. Figure 4.26 displays the CPU Registers’ contents in cycle 1. ViewRamAfterRun button displays the Ram content after code execution.

![CPU Registers In Cycle 1 Using Run Button](image)

**Figure 4.26: CPU Registers In Cycle 1 Using Run Button**

Button Run+Ram allows the Instructor to run the Hex code and see the CPU Registers’ contents in each cycle before and after fetch, and after execution and the CPU content after the related command execution by clicking the button with associated number. Figure 4.27 displays the CPU Registers’ contents in cycle 1. ViewRamAfterRun button displays the Ram content after code execution.

It is important to mention that if for any reason the Instructor hits RunToEnd, Run, or Run+Ram button without loading the RAM, the RAM will be loaded with a default Hex code which will be displayed in the Loaded Ram textarea. Then this default Hex code will be executed and the result will be available to see by clicking the key associated with each cycle visible with the cycle number.
4.9.4 Left Center Frame
This area contains a dropdown list of the available assembly language commands as reference allowing the Instructor to edit, delete, or create new commands. By selecting a command from the list and clicking Edit or Delete button, the Instructor is able to perform selected operation on the selected command. The Instructor is also able to create a new command to be used by students during writing assembly code just by using New button provided and editing the CPU class as discussed before.

Selecting a command from the command list and clicking the Edit button directs the Instructor to the command edit page allowing the Instructor to edit any desired field(s) of the command. Figure 4.28 displays the command edit page. After applying the desired changes to desired field(s), the Instructor can select Submit button to save the changes or the Go Back link to dismiss the changes. However, it is important to mention that the opcode must be unique and one opcode cannot be used for two different commands. If the Instructor tries to change the opcode to the opcode associated with another assembly command, the system will not allow the save operation to complete and returns a message to inform the Instructor that the opcode he/she is trying to use is already used for another command, demanding him/her to select an opcode not used before. Figure 4.29 displays this warning message. The Reference Page link returns the Instructor to the Assembly Reference Page while the left-arrow on the top of the screen directs the Instructor back to the command edit page.
Figure 4.28: Assembly Command Edit Page

Figure 4.29: Opcode Duplication Warning Error
Figure 4.30: Assembly Command Delete Page

Figure 4.31: Assembly Command Create New Page
Figure 4.30 displays the command delete page asking the user for delete confirmation. If the Instructor clicks the Confirm Deletion button, the selected command will be deleted from the reference list. However, the Instructor can decide not to delete the selected command and use the Go Back link to be directed to the Assembly Reference Page.

Figure 4.31 displays the command create new page allowing the user create a new command. If the Instructor clicks the Submit button, the selected command will be added to the reference list if the inserted opcode is not duplicated. However, the Instructor can decide not to create the new command and use the Go Back link to be directed back to the Assembly Reference Page.

4.9.5 Right Center Frame

This area contains a dropdown list of the students’ names allowing the Instructor to edit or delete the login and/or the password of the selected student using the Edit or Delete button or to create new login and password for a student not in the list using the New button.

Selecting a student from the list and clicking the Edit button directs the Instructor to the edit page allowing the Instructor to edit the login or password of the selected student. Figure 4.32 displays the Login/Password edit page.
After applying the desired changes to the login and/or password, the Instructor can select Submit button to save the changes or the Go Back link to dismiss the changes.

![Login/Password Delete Page](image)

Figure 4.3: Login / Password Delete Page

Having selected a student from the list, if the Instructor clicks on the Delete button, the Instructor will be directed to the login/ password edit page displaying the student information with the Confirm Deletion button. If this button is clicked, the student will be removed from the list. However, clicking the Go Back link will dismiss the delete operation directing the Instructor back to the Username/Password page. Figure 4.33 displays the Login/Password Delete Page.

The New button in this frame allows the Instructor to create a new account for a student not in the student list. Clicking this button directs the Instructor to the create new account page with textboxes designed to insert CSUNID, First Name, Last Name, Login and Password. After completing all these fields, the Instructor can save the new account in the student list using the Submit button designed for save purpose. However, clicking the Go Back link will direct the user with Instructor role back to the Username/Password page. Figure 4.34 displays login/password create new account page.
Figure 4.34: Login / Password Create New Page
Chapter 5

Available Features for Student

If a user with role Student successfully logs in, a success message will appear along with a link named Student page that will direct the user to the DBA Control Panel Page. Figure 5.1 displays this page.

![Log In Page](image)

**Figure 5.1: Student Success Message**

By clicking on the student page link, the user will be directed to the Student Page where he can select between features available for role Student. A Student can either go to the student assembly page using Go to the Assembly-Student Page or check his scores using Check Your Scores link. Figure 5.2 displays this page.
5.1 Student Assembly Page

Figure 5.3 illustrates the Student Assembly Page (Assembler Editor Page). Using the facilities provided in this page, the student can extract a homework assignment from the homework assignments list, provide a code as solution to the selected assignment, run the solution code through the system provided in this page and evaluate the result to make sure the solution provides the desired results. Then, the student can submit his/her solution by clicking on the save button. The student can also edit his/her previously submitted solution and resubmit the solution. In the following you can find detailed explanation of what exactly can be done in this page.
5.1.1 Top Center Area

This frame allows the student to select an assignment from the dropdown list and determine it as selected assignment by clicking the Edit Solution button. If the student has already submitted a solution for this assignment, then, his/her solution will be displayed in the left textarea of the center frame so that he/she can run or edit the solution. The Go Back link presented in this frame will direct the student back to the Student Page. The student can use this link to go to the student page to log out. Figure 5.4 displays a case where the student has submitted a solution to an assignment and he selects that assignment to edit the submitted solution. As you can see in this Figure, the solution is displayed in the left textarea in the center frame.

However, the student does not have to select any assignment from the assignment list in this area if he/she just wants to practice or to learn how to use this system. The student can write a sample code in the left textarea in the center frame and follow the other steps to run this code. But it is important to know that if no assignment is selected, the code in this textarea cannot be saved. Also, if the student wants to submit a solution to an assignment, it is important to first select the assignment from the list and click the Edit Solution button to let the system know which assignment is going to be worked on. Otherwise, if the student writes the code first and then selects an assignment and hits the Edit Solution button, the code will be gone. This means the order really matters here.
5.1.2 Left Center Area

This area which is distinguishable in the page as Assembly Reference frame, allows the student to find required information about each available command he/she can use in the solutions. The commands are sorted alphabetically in the command list. Selecting a command from the dropdown command list in this frame and clicking on the Display button, will display the information required to know about this command. Figure 5.5 illustrates the information table of add command in the bottom frame.

The information table contains the following fields: Opcode, Register, Meaning, Addr. Modes (Address Modes), Status Bits, and Example. The Opcode is the representative of the command in binary system and is unique for each command. The Meaning field determines what this command does when it is used in the code. Addr. Modes is the command address mode. The Status Bits field shows if the status can be zero or not. NZ means the status cannot be zero whereas Z indicates the status can be zero. And Example field displays an example of how to use the selected command.
5.1.3 Center Area

The left textarea is designed for coding. The student types his/her code in this textarea to run it through this system. The Assembler button converts the assembly code into its equivalent in Hex code/numbers. The result will be then displayed in the right textarea. Figure 5.6 displays the Hex code in the right textarea.

The Load button loads the Ram with the Hex code to prepare it for run. The Hex code then will be displayed in the Loaded RAM textarea in the bottom frame. This field keeps the Hex code without any changes after code execution so that the student is able to compare the contents of the Ram before and after code execution. Figure 5.7 displays the Loaded Ram textarea with the Hex code loaded.

Oct button allows the student to convert the Hex code/numbers displayed in the right textarea to its octal equivalent and display the result in the textarea located on the bottom of this frame. Figure 5.8 displays the octal code in the bottom textarea.

Bin button allows the student to convert the Hex code/numbers displayed in the right textarea to its binary equivalent and display the result in the textarea located on the bottom of this frame. Figure 5.9 displays the binary code in the bottom textarea.
Figure 5.6: Hex Code In The Right Textarea

Figure 5.7: LOADED RAM With The Hex Code Loaded
Figure 5.8: Octal Code In The Bottom Textarea

Figure 5.9: Binary Code In The Bottom Textarea
5.1.4 Right Center Area

This area named as Manual gives a short direction to the student to know how to use the facilities provided in this page. It is just a simple description of buttons and textareas available to use.

5.1.5 Right Center Area

Loaded RAM textarea keeps the Hex code to execute. The Input textbox on the right is available to the student to set the input if necessary. The output textbox on the right displays the code execution output if there is any. Between input and output textboxes there are three buttons for running the Hex code located in the Loaded RAM as follow: RunToEnd, Run, and Run+RAM.

Button RunToEnd allows the student to run the Hex code and see the CPU Registers’ contents in each cycle by clicking the button with associated number. Figure 5.10 displays the CPU Registers’ contents in cycle 1.

The ViewRamAfterRun button displays the content of the RAM after code complete execution. Figure 5.11 displays the Ram content after code execution.

The CPU with its ten registers displayed at the bottom of this area contains the final values of each register in the associated textboxes. If you compare the content of these registers with the value of the equivalent registers presented in the last cycle, they should match. Figure 5.12 displays the CPU registers’ contents of the last cycle to compare with the equivalent registers displayed at the bottom of this area.

![Image](image_url)

Figure 5.10: CPU Registers’ Contents In Cycle 1 Using RunToEnd button

64
Figure 5.11: Ram Content After Execution Using ViewRamAfterRun

Figure 5.12: CPU Registers’ Contents In Cycle 10 Using RunToEnd button

65
Button Run allows the Student to run the Hex code and see the CPU Registers’ contents in each cycle before and after fetch, and after execution by clicking the button with associated number. Figure 5.13 displays the CPU Registers’ contents in cycle 1. ViewRamAfterRun button displays the Ram content after code execution.

![CPU Registers' Contents In Cycle 10 Using Run button](image)

**Figure 5.13: CPU Registers’ Contents In Cycle 10 Using Run button**

Button Run+Ram allows the Student to run the Hex code and see the CPU Registers’ contents of each cycle before and after fetch, and after execution and the CPU content after the related command execution by clicking the button with associated number. Figure 5.14 displays the CPU Registers’ contents in cycle 1. ViewRamAfterRun button displays the Ram content after code execution.

It is important to mention that if for any reason the Student hits RunToEnd, Run, or Run+Ram button without loading the RAM, the RAM will be loaded with a default Hex code which will be displayed in the Loaded Ram textarea. Then this default Hex code will be executed and the result will be available to see by clicking the key associated with each cycle visible with the cycle number.
5.2 Check Your Score Link
Clicking on this link directs the student to the My Score Page allowing the student to see his scores received in all the active assignments available to the student to work on. If the student has not submitted any solution to any assignments or the solution to an assignment is not yet graded by the instructor, the score will be displayed as 0 in the score field associated with that assignment. Figure 5.15 displays the scores achieved by student John Smith. The Go Back link will direct the student back to the Page.
Figure 5.15: Scores Table Of Student John Smith

<table>
<thead>
<tr>
<th>Homework</th>
<th>Student Solution</th>
<th>Score</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversion</td>
<td>load IR, store IR, load IR, add IR, store IR, load IR, add IR, store IR</td>
<td>3</td>
<td>2012-10-18  13:45:47</td>
</tr>
<tr>
<td>NextTable</td>
<td>load IR, store IR, add IR, store IR</td>
<td>4</td>
<td>2012-10-18  13:55:50</td>
</tr>
<tr>
<td>Allocate</td>
<td>load IR, store IR, add IR, store IR, load IR, load IR, store IR</td>
<td>1</td>
<td>2012-10-18  12:02:49</td>
</tr>
<tr>
<td>NextBlock</td>
<td>load IR, store IR, add IR, store IR</td>
<td>3</td>
<td>2012-10-18  13:06:52</td>
</tr>
</tbody>
</table>
Chapter 6

Conclusion

The software package of the thesis is Assembly Editor, a web based software with a support of database and Object Oriented Programming method. Assembly Editor is designed on a 3-tier system architecture: Web Server, Data Base Management System (DBMS), and Web Browser. MySQL is the DBMS and PHP is the programming language that are used in this project. The functionality of the software is based on a framework which enables the Assembly Editor to operate through interaction between index.php file located on the Server and a database table that stores code behind webpages in records using some classes located in another database table.

The main purpose of creating Assembly Editor Software is to assist students in learning Assembly programming concepts. This software helps the students write and run Assembly programs displaying the code execution results through simulating the CPU and the Memory behavior before and after executing each Assembly command.

However, unlike other learning tools, Assembly Editor is not only a learning tool. It is also a powerful software that allows the instructor to provide the students with homework assignments making them available through internet and enables the students to submit their solutions conveniently using any computer from anywhere any time before assignments’ due date. The instructor can then easily assign grades to the students’ solution to each homework assignment by fetching them from the database, running them and submitting the grades. Furthermore, the instructor is provided with the facility of tracking homework assignments’ grades and students’ progress. And finally, the student is able to see his grades in a table along with homework assignments’ description and his submitted solution to each assignment. He can use this table to evaluate his performance and progress towards Assembly Language programming.

6.1 Future Work

Assembly Editor is an easy to use software that enables the instructor to provide the students with Assembly Language programming related homework assignments, and the students to submit their solutions along with programming assist and a system to simulate the CPU and the RAM before and after each command execution.

A future work can be extending the software to provide the instructor with the ability of creating online exams making them available to the students within a specific time period and the students to submit their solutions within the specified time period. Also, the software can be improved to compare the submitted solutions to the instructor’s solutions and assigning grades to solutions. This software is designed to work only for the students of one class. But, can be easily modified to support many classes instead of only one.

Moreover, this software can be improved to a 3D illustrative and a more colorful software for a better visualization. This software is heavily table driven requiring permanent access to the database tables during processes all the time as except the index.php file which is located on the Web Server, the whole classes and the GUI codes are stored in the
database tables. This might slow down the processes specially when users grow in number. Thus, a more flexible environment like “.net” framework may be more suitable for implementing Assembly Editor similar softwares. JAVA environment can also be considered as a better choice due to its class- base and object-oriented nature.

And, it covers only Assembly Programming Language. A similar work can be done for other programming languages to reduce the learning time and increase the programming interest among students. Also, any processes from science and engineering to office management applications can easily be developed and implemented by extending or applying new features to the Assembly Editor.

The concept of the Assembly Editor software, regardless of the operating platform, is so flexible and robust that makes the development and implementation of different applications/models reasonably easy and inexpensive.
References


