|  |  |
| --- | --- |
| A building with a sign on it  Description automatically generated  MULTIPLICATION GAME  SE2340-501 Computer Architecture | **Abstract**  Use your multiplication facts to beat the computer. Enter your input to choose your number. You and the computer take turns moving one marker at a time. Get 4 in a row before the computer and you win.  **Created by**  Aaron, Momo, Sean, Sujin |

**1. About this program**

1. **A description of the program**

The Multiplication Game in MIPS Assembly is a two-player strategy game played on a 6x6 board. The game involves taking turns between a user and the computer, with the objective of creating a line (horizontal, vertical, or diagonal) of products on the board. The products are obtained by multiplying two numbers chosen by the player and computer.

The game begins with the board being initialized from a set of numbers into a space array so that they can later be replaced with characters. Next the computer moves the top slider to a number between 1 and 9. Next, the program prompts the user to input a letter, ‘b’ or ‘t’, to pick a slider, and number in the range of 1 to 9. The program validates the input to ensure it has not already been taken by either the user or the computer. If the space is available, the number is replaced in the array with the corresponding character ‘o’ for player and ‘X’ for computer. After the character is placed in the array, the program checks every row, column, and diagonal to see if it can find a sequence of 4 characters in a row belonging to either the player or computer. After the check for a win is finished the game swaps turns by updating the register $t6 to 0 for player turn or 1 for computer turn. During the computer it randomly selects a number either 0 or 1 to pick either the top or bottom slider and then randomly picks a number 1-9 to move that slider to. If the numbers, the computer picks are not available it will re-randomize the numbers it picked until it finds an available space. This process will repeat until the program finds 4 in a row of either the player or computer character. After which it will jump to either a victory or defeat message and prompt the player to play again or exit the program.

1. **The challenges that you and your team had and how did you or the team overcome them.**

The biggest difficulty I encountered while working on this project was completing the project by using MIPS, an assembly language. I was already familiar with the language, but implementing the game using a new language, especially MIPS, required a completely different syntax, and the limited number of registers led to me making frequent mistakes.

Another difficulty Momo had to work through was that this project was mainly about figuring out how, from an analysis point of view, to break the game down using the divide and conquer concept effectively to ensure that when the game is broken into smaller parts, each part holds a similar amount of significance.

1. **What you have learned by doing the project.**

This project afforded me a comprehensive understanding of the MIPS assembly language, enabling me to proficiently employ its diverse applications. Through the composition of MIPS code and the meticulous design of code structures, I have successfully exercised fundamental programming principles. The development of game logic and algorithms highlighted the importance of a methodological approach to data collection, and emphasized the calculation of data storage addresses as opposed to an array index access.

The primary takeaway from this endeavor are the insights gained into effective team project management and the benefits of team contributions. The team formation, role designations, agendas, and code refinement are clearly important to achieving a good result. This experience also emphasized the significance of consistent communication with open exchanges of ideas and criticisms.

To summarize, this project has not only furthered my technical skills, but has also given me more experience in project management, teamwork, and effective communication.

1. **A discussion of algorithms and techniques used in the program, e.g. how to generate and display the cards? How does the program work?**

The first thing the program does is loop the values from a word array into a space array so that they can later be updated with characters. It does this by indexing both of the arrays with the same register ($t1) so that they stay aligned throughout the entire initialization. It then also prints the slider by printing a series of spaces depending on what number the slider is meant to be pointing to along with a string of the numbers it can be pointing to. Next, the program prompts the player for the character ‘b’ or ‘t’ and if it does not find the values 98 or 116 through a beq statement it will re-prompt the user for the characters again. Similarly when it prompts the player for a number if it is not a number 1-9, checking for any numbers greater than 10 or less than 0, it will re-prompt for a new number. It then loops through the array to see if the numbers multiplied are available within the array. It does this by checking to make sure it does not reach a number greater than the number input, using beq statements to avoid the characters for player or computer (equal to 111 and 88). If it finds the requested number, it will replace it in the array using the same index from the loop to ensure it gets placed in the correct spot. Finally, the program loops through the array again to search for sequences of player or computer characters. It does this by searching each row, column, and diagonal for either 88 or 111. When it finds one of those characters it increments $s4 by 1 and if it finds any other character it resets $s4 to 0. It continues this until we eventually reach a series of 4 characters in a row in which we immediately jump to the end of the program. This then prompts the user for the character ‘y’ or ‘n’, the value 121 jumping back the beginning of the program and resetting all the registers to 0, and the value 110 jumping to the exit of the program.

1. **Contributions of each team member (peer evaluation),**

Aaron:

Assignments - Game Logic.

Contributions – Creation of version of game logic, chat group member.

Momo:

Assignments - Board Object, User Manual.

Contributions - Planning, User Manual work, created a board object and User Manual versions, active in chat group communications, provided suggestions and support.

Sean:

Assignments - Computer Object, Coding.

Contributions: Provided finalized versions of User Manual, board & computer objects, User Control version, active in chat group communications, provided suggestions and support. Responsive, dedicated, and active participation, timely work submissions.

Sujin Lee:

Assignments: User Control, Video

Contributions – Created and edited video, created this report, created a version of User Control, active in chat group communications, provided suggestions and support.

**2. A short video clip demonstrating the program in action.**

[Click to access video](https://youtu.be/-no7ciZKa-8)

**3. All MIPS assembly language modules that are needed to run your program.**

Computer\_Score.asm, Multiplication\_Game.asm, Player\_Score.asm, Reprint.asm

**4. A user manual on how to run and how to use the program.**

[Click to access User Manual](https://sujinbonnaud.com/User_Manual.pdf) (or user\_manual.pdf in the zip folder)