Arshveer Singh
CS 2340.502
Professor Alice Wang

Project Report

a) This program is a math-matching game created in the MIPS Assembly language. There is a grid consisting of 4 rows and 4 columns. In each grid, there is either an expression, such as ‘4x2’, or a number, which is the result of an expression, such as ‘8’. These values are hidden behind an asterisk. It is the player’s responsibility to unhide the cards and match them accordingly. This is done by the game asking the player to input the row number and column number of a card. For the row number and column number, the player can input 1, 2, 3, and 4. If they put 0, or any number that’s 5 and above, they will get an error, telling the player to input a correct number. If the cards match, they will stay revealed since they have already been matched and a chime sound will play using the MIDI mips sound system. The sound system utilizes the built in sounds and each block printed is a note from the system. If the cards do not match, they will stay hidden. There are a total of 8 pairs that need to be matched. After the player finishes, the game will congratulate the player for winning, and a sound will be played as part of it. The sound is supposed to be a fanfare. The fanfare uses a trumpet as the instrument. This is part of the built-in sounds in MIPS. This sound uses four notes (four blocks of code) that are played with rest blocks to make the sound flow smoother. The game will ask them if they wish to play again or exit the game. If the player wishes to play again, the values in the grid will be reset to asterisks. If they do not wish to play again, the game will gracefully exit.

b) We encountered many challenges during the development of the game. One of these challenges was properly displaying the multiplication problems onto the grid which was not showing due to the grid’s cell size being small. Another challenge we encountered was the coordinate system not linking to the correct cell, for example the coordinate (1,1) would be assigned to the cell that was supposed to be (3,2). To fix this issue, we restructured our coordinate system to use a multidimensional array which would read the grid from left to right and correctly assign each coordinate to the right cell. One other big challenge we faced was the implementation of the sound system. The original sound system I used attempted to use an imported external .wav file that played music when the user won the game or got a pair, but this method was extremely difficult to implement. Instead, I opted to use the built-in MIDI sounds that mips offers which worked seamlessly with our code and didn’t cause any further issues. Despite these challenges, we learned to improvise from our mistakes and built a game that was easy to debug, very smooth, and very fun to play.

c) I did learn about how sound (MIDI) can be implemented into MIPS programs. I learned about how timers can be implemented. I learned about how modularity is an extremely good practice when writing a large-scale program like this. I learned about how stack pointers manage memory. It gave me a good opportunity to understand the last-in, first-out nature of the stack data structure.

d) One technique we did use for the game was ensuring modularity. Files responsible for displaying the grid, handling the timer, handling the game logic, and more are kept in individual files. By doing this, we were able to easily make changes to a file if the program was not working as intended, and we were able to easily implement new features in the game. We used a multidimensional array, which in this case, is 4 by 4, to create the grid that the game uses. The game reads cells from left to right in each row. An example of this would be that if the player inputs row 1 and column 1 for the coordinate, the game picks the cell at the top left corner. This game uses stack pointers to store the address of the last element added to the code and in this case, we used stack pointers to store the user-inputted coordinates. After the coordinates are evaluated to see if there's a match, the addresses are removed from the stack pointer and then assigned to the next user-inputted coordinate. This method of storing memory makes the game smoother and more efficient to run, causing less problems in the long run.

e) My partner has implemented the sound that the program uses. He has also implemented the stack memory functions that the game uses. He has implemented the timer and the counter that shows how many unmatched cards are remaining. Also, he has implemented the logic that is responsible for the user input, and input checking. He has implemented the logic that reveals the specified grid when the user selects it.