Description of Program:

Multiply Easy is a memory game where the player matches tiles based on the tile's multiplicative product. All tiles start out "face down". Only two tiles are flipped at a time, then both are flipped back if their products do not match. Upon a match, those cards are permanently displayed, and the player must go and try to match the rest of the cards. Upon matching all the cards, the player will beat the game.

Challenges:

Throughout our development of the game, we struggled with a lot of things. The first being trying to get files to compile together which took a lot longer than we thought (a few days). After that we ran into the issue of the graphics not displaying the way we wanted them to, so we settled for a simpler grid approach. I personally had an issue with the timer in terms of how to calculate the time accurately and when to display but I did my research and eventually figured it out. To solve our problems, we just needed to put more time and effort into, which we did towards the end.

Things Learned:

I never liked programming in low level languages but by doing this project I grew to like MIPS. It wouldn’t be my language of choice but if I had to use MIPS, I would feel comfortable using this language. Learning how to work with others on a complicated project like this was also a useful skill picked up. Very glad my partner was patient enough with me throughout the whole process, as I with him. This experience has overall benefited my ability to program efficiently and in a deadline.

Discussion of Techniques:

To display the board, we had a loop that revolved around getting the user input. Once grabbing user input (and checking if it is valid input), we would print out a predefined ASCII board, with the not revealed cards covered with a "?". To track revealed cards, we would have a check\_match function that would update the revealed value of each card, and upon displaying the board, said revealed value would determine if a "?" is printed or the actual card text is printed. To describe the program from the top, first, the prompt message asking the player to play is printed, followed by a user input and input validation of said input. If the user input is incorrect, then that process will loop. Otherwise, the game will proceed. The game then loads the board values and prints a starting "Go!" Message. Then, the game will put needed registers into the stack for future use and begin to initialize the expressions array and counter and store them into memory. After that, the program will shuffle the cards in the array using the randomizer module to change up the cards so that every game is different. The game officially starts after this. The main loop goes as follows. First, get user input to see which cards they want to flip, and validate that input. Second check if the cards provided match card value, which is the "cards" value in .data. If the cards match, increment the number of matched pairs and display the new board and message. If they don't match, temporarily display their values then hide them once again. Repeat this entire main process until the number of matched pairs equals 8, then ask to replay the game.

Contributions of Partner:

A majority of our project was spent working together; very rarely did we split up on things. The only things we did individually were the randomizer (Owen) and the timer (me). We spent most of our time debugging the logic module since it was the most important for the game functionality. The graphics were also a bit difficult to align properly and have everything displayed well but we managed to get the hang of it. Overall, Owen was an incredible partner to work with and I would gladly work with him in the future.