Mind Reader

|  |  |  |  |
| --- | --- | --- | --- |
| Course: | CS3340.501 | Professor: | Nhut Nguyen |
| Team: | The Shield | Submitted by: | Chaoran Li |
| Team Members: | Chaoran Li, Jiaer Jiang, Xue Cheng, Pengfei Tong | | |

# a) Description

The team project this semester is the Mind Reader game. The game asks the player to think of a number between 1 to 63, generates and randomly displays six cards that contain 32 numbers at a time. The player can input ‘y’ or ‘n’ based on whether the guessed number been displayed on the card. After six rounds, the correct number is displayed. The program can get input from players and decide whether it’s valid. If the input is invalid, a warning tone will be for announcement. The user can play the game repeatedly or choose to end the game immediately. Also, we did extra features including the graphic display, background music, sounds to indicate invalid input, and shuffled cards for more deceptive display.

# b) Challenges

Basic requirements part:

1) Low Readability for MIPS and MARS

As an assemble language, MIPS requires more time and effort in reading new codes and debugging. The lack of common standards and practice made things worse.

2) The Order of Displayed Cards must be Random

The “random” means that we need to generate numbers by shuffling, which was not provided by the existing syscall. We wrote a shuffle method based on a syscall 42 and a popular shuffle algorithm.

Extra credits part:

3) Graphic/Color Display

Three choices for visualization design: i) Dialog syscall; ii) Bitmap Display Tool; iii) Rewritten MARS for new syscall. We designed the game window based on Bitmap.

4) Sound and Music

Two plans for music: i) Tone syscall; ii) Rewritten MARS for new syscall. We chose the first one which will play tones based on input from txt. file.

# c) What I have learned

1) ***Assemble Language and Bitmap Display***

I learned how to write the assembly language program, apply the instructions to develop the game and draw in a bitmap display. Also, I learned we can implement the tools in MARS to help us achieve the features we expected.

2) Teamwork

I learned how to cooperate with team members to complete our projects efficiently. We divided the project into different parts and assign the tasks for each member. We met twice a week after the class to report the progress of the task and talk about the issues we had. We did integration testing together to make sure all features working properly.

# d) Algorithms and Techniques

1) Main Algorithm

For input should be minimum, we use the binary search here. Integers from 1 to 63 can be expressed in 6-bit binary, and we can find the answer if we get each bit of it.

Below uses the original Java Code for the main function to explain our algorithm: (randomly displaying card had been added in the later version)

**import** java.util.Scanner;  
  
**public class** Solution {  
 **public static void** main(String[] args) {  
 **int** res = 0;  
 **int** bit = 1;  
 Scanner input = **new** Scanner(System.***in***);  
 **for** (**int** i = 0; i < 6; i++) {  
 **int** num = bit;  
 **for** (**int** j = 0; j < 32/bit; j++) {  
 **for** (**int** k = 0; k < bit; k++) {  
 System.***out***.print(num + **" "**);  
 num += 1;  
 }  
 num += bit;  
 }  
 System.***out***.println();  
 String s = input.nextLine();  
 **if** (s.equals(**"y"**)) res += bit;  
 bit <<= 1;  
 }  
 System.***out***.println(**"the number is: "** + res);  
 }  
}

The main function can be summarized as a three-layer loop. The outer layer displays 6 cards in turn, and the inner two layers are used to generate the numbers on each card. Each card will select one of the 6 bits as 1, and then the inner double-layer loop will traverse other digits to get all numbers. When the input for one card is y, the corresponding bit is 1, otherwise, it is 0. Combining with the relationship between digit and card, we can easily get the final number.

2) Special Case

Two special cases are discussed here.

One is that the player enters invalid characters. We will prompt a warning sound until we get the right input.

The other is that the player thinks of a number other than 1-63. At this time, the input is ‘n’ six times. We will prompt the number is out of range, and that is why 0 (input is ‘n’ six times) is excluded from the numbers.

3) Shuffle Function (Ask for extra credits for shuffling numbers on each card)

Original algorithm:

**import** java.util.Arrays;  
**import** java.util.Random;

**public class** Solution {  
 **public static void** main(String[] args) {  
 **int**[] a = **new int**[20];  
 **for** (**int** i = 0; i < a.**length**; i++){  
 a[i] = i;  
 }  
 System.***out***.println(Arrays.*toString*(a));  
 Random rand = **new** Random();  
 **for** (**int** i = 0; i < a.**length**; i++){  
 **int** j = rand.nextInt(a.**length** - i);  
 **int** tmp = a[i];  
 a[i] = a[j];  
 a[j] = tmp;  
 }  
 System.***out***.println(Arrays.*toString*(a));  
 }  
}

The function will traverse words in the specified memory area. For each word, use syscall 42 to choose a random word after the word and swap them. After that, all words are shuffled.

Using the shuffle function, we can display cards randomly. We can also shuffle the numbers on different cards. In the old version, we also include randomness in whether the chosen digit is ‘0’ or ‘1’. However, this will cause a logical error in judgment when the number is not between 1 and 63. So we exclude it in the later version.

4) Print Pictures in Bitmap (Extra credits)

We used the bitmap pixel display, use the pixel to draw number cube 45 x 225 pixel and word 8 x 16 pixels for digits from 0 to 9. We are able to call specific digits when we need to display it. Additional graphical display background is made and adjusts to the proper pixel size to fit in the bitmap. Different graphical picture content game information that asks the game player for user input to make the game flow. Modified mars added Syscall 60 that made inserted picture background displayed properly.

5) Ringtones and Music (Extra credits)

**Background music**

The purpose of the read number notes added in the program is to make the computer beep according to a text note file that is named 'Music.txt'. The first line in the text file is a three-digit number that corresponds to the number of notes going to be played, from 001 to 100. For each note, it represents a different pitch and instrument tone. It knows the notes D, E, F, G, B, C, and C #. The sequence of different notes made up background music been inserted after the first and last graphical background is displayed. In the assembly code, Syscall 33 used that plays a note and waits to continue until after the beep.

**Ringtones**

The ringtones are carried out using the MIDI out system call by MIPS. Different pitches and instruments were used to distinguish between the ringtones of ‘y’ input, ‘n’ input and other invalid input.

# e) Contributions (Peer Evaluation)

***Chaoran Li:***

* Write the algorithm for the random shuffle
* Lead the coding in MIPS
* Integrate all the modules and functionalities
* Test and debug the program
* Participate in the project discussion
* Help to write the documentation

***Yuer Jiang:***

* Write the main algorithm in java
* Lead the coding for graphic display by bitmap
* Test and debug the program
* Participate in the project discussion
* Help to write the documentation

***Pengfei Tong:***

* Lead the coding for background music and sounds to indicate invalid input
* Create the background music
* Test and debug the program
* Participate in the project discussion
* Help to write the documentation

***Xue Cheng:***

* Record the short video demonstrating the program
* Write the user manual
* Test and debug the program
* Participate in the project discussion
* Help to write the documentation

# f) Suggestion

This project is a great practice for us to apply MIPS instructions that we learned in class and cooperated with team members to explore the extra features of MARS. I hope we can have multiple games to choose or define the topic by ourselves.