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

Mind Reader would guess the secret number according to player's choice. The program will first ask player to think of a number between 1 and 63. Then six cards will display with 32 numbers on each card. Player will be asked to input ‘y’ or ‘n’ based on whether the secret number is in the card. After the last card displayed, this program will find the answer according to player’s choice. At the end of a game, the program will ask player if he/she wants to play another game and then repeat or end the program.

# b) Challenges

Minimum 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

Obviously, the essence of “random” here is shuffle, which was not provided by existing syscall. We wrote a shuffle method based on syscall 42 and 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 according to input txt file.

5) MARS rewritten

We had tried to rewrite MARS, but it was late for us to realize that this solution. In order to optimize the background picture display in Bitmap in the later period, we used a MARS version rewritten by others for extra syscall.

# c) What I learned

# d) Algorithms and Techniques

1) Main Algorithm

For input should be minimum, we use 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 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 inner double-layer loop will traverse other digits to get all numbers.

When 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 a right input.

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

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

4) Print Numbers in Bitmap (Extra credits)

(yuer)

5) Print Pictures in Bitmap (Extra credits)

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

6) 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 represent a different pitch and instruments tone. It knows the notes D, E, F, G, B, C, and C #. The sequence of different notes made up a background music been insert after the first and last graphical background are 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.

/\*\*

**1) Algorithms**

The mind read game seem magic but it has a very simple algorithms behind it. Computer displays six sets of numbers. For every i from 0 to 6, it displays the numbers with the same (either 0 or 1, selected randomly) binary digit in the i-th position counting from right. Answering "Yes" the player actually tell the program whether the digit is 0 or 1. Thus digit after digit, the program collects the valid information about the binary representation of the number that player had in mind. Since (111111)2 = 64 > 63, any number below 64 needs at most 6 bits to be written in the binary system.

For example, assume game player choose number 38. Its binary representation is 100110. I have to guess 6 binary digits. All numbers split into two groups: those that have the rightmost digit 0 and the rest that have the rightmost digit 1. Number 38 belongs to the first group. The program randomly select which group (below 64) to display. The player answer will be either "Yes" or "No", depending on which of the two groups actually shows up. But, regardless of which group is shown, once you press one of the buttons, The program can determine the rightmost digit of your number. With 38 in mind, if you see the first group, you will press the "Yes" button. If you see the second group, your response will be "No".

Other digits are determined in the exactly same manner. All numbers split into two groups: those that have their 3rd (from the right) digit equal to 0 (1, 2, 3, 8, 9, 10, 11, 16, 17, ...) and the rest that have 1 in the 3rd position (4, 5, 6, 7, 12, 13, 14, 15, 20, ...). The program select one of the groups randomly. Player answer will depend on which of the two groups has been selected. But regardless, pressing one of the buttons player tell the program which digit it is. For 38, if program display the first group, the answer must be "No". For the second group, the answer must be "Yes." In both cases we received the information to realize that the 3rd digit of your number is 1. Same thing, after 6 input we able to reveal the 6 binary digits which is the specific decimal number.

**Extra credit graphical display**

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

**Extra credit 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 represent a different pitch and instruments tone. It knows the notes D, E, F, G, B, C, and C #. The sequence of different notes made up a background music been insert after the first and last graphical background are 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)

# f) Suggestion (Optional)