Bahria University,

Karachi Campus



COURSE: Computer Architecture and Logic Design

TERM: FALL 2022, CLASS: BSE- 3 (A)

PROJECT NAME

HANGMAN GAME IN MIPS

Engr.Dr.Sorath Hansrajani/ Engr. Ramsha Mashood

Signed Remarks: Score:

**GROUP MEMBERS:**

* **KASHAN RIAZ (02-131212-075) (Group lead)**
* **EIMA NASIR (02-131212-020)**
* **HASEEBULLAH (02-131212-069)**

1. TABLE OF CONTENT

* Introduction and Problem------ Pg 3
* Technology------Pg 3
* Functionalities------Pg 3
* Module Distribution---Pg 4
* Code-------Pg 4
* Interfaces-----Pg 46
* Conclusion----Pg 49

1. INTRODUCTION & PROBLEM

In less than 50 years, video games have emerged from obscurity and become one of the fastest-evolving forms of popular culture. Our goal is to produce a playable and intuitive video game for the general population. Attempting to guess words in the classic paper and pencil game of hangman is only allowed a certain number of times. To give it a newer, more contemporary style and to educate the younger generation to this traditional game, we felt it was time to turn it into a video game.

1. TECHNOLOGY

The Hangman Game is made in assembly language using MARS MIPS IDE. The game runs and displays in the console. The console is used to take user inputs and display the outputs to the user. A word bank (according to categories) is made within the .data section of the code which is used to randomly pick a word and give it to the user to guess

1. FUNCTIONALITIES

Logic Module Functionalities:

* The user is prompted to enter characters.
* Giving the user the choice of a word-related hint.
* 7 tries are given while guessing a single word after which the game is over and user loses.
* The user can play another game if they wish to when they’re asked if they want to play again when a game ends.

Design Module Functionalities:

* The hangman drawing is made step by step when each wrong alphabet is entered.
* Designing the hangman
* The hangman is made in a particular sequence.
* When the hangman is fully drawn, the game is over and user is asked whether they want to play again. A reset hangman drawing is started in a new game.
* If the user wins, the hangman will be seen standing.

1. MODULE DISTRIBUTION

|  |  |  |
| --- | --- | --- |
|  | Name | Module Distribution |
| 01 | Haseebullah (02-131212-069) | Game logic and implementation. |
| 02 | Kashan Riaz (02-131212-075) | Game logic and implementation |
| 03 | Eima Nasir (02-131212-020) | Gamic logic and Design |

1. CODE

.data

#words

Country: .asciiz "China"

Animal: .asciiz "Hyena"

Yacht: .asciiz "Yacht"

Earth: .asciiz "Earth"

Color: .asciiz "Black"

#messages

message1: .asciiz "\nWelcome to Hangman.\n"

message2: .asciiz "\nYour word length is: \n"

message3: .asciiz "\nWrong guess! Guess again!\n"

message4: .asciiz "\nThat guess is correct!\n"

message5: .asciiz "\nYou lose!The death penalty awaits.\n"

message6: .asciiz "\nCongratulations!You win!Your score is: 1\n"

message7: .asciiz "\nMake your guess please.\n"

message8: .asciiz "\nWould you like to play again?[y/n]\n"

message9: .asciiz "\nWould you like to continue?[y/n]\n"

message10: .asciiz "\nCongratulations!You win!Your score is: 2\n"

message11: .asciiz "\nCongratulations!You win!Your score is: 3\n"

message12: .asciiz "\nCongratulations!You win!Your score is: 4\n"

message13: .asciiz "\nCongratulations!You beat the game!\n"

message14: .asciiz "\npppoopoo\n"

#hints

CountryHint: .asciiz "\nHint: Country that is famous for a certain wall\n"

AnimalHint: .asciiz "\nHint: Animal which was the secondary villain in the lion king\n"

YachtHint: .asciiz "\nHint: A very fancy boat\n"

EarthHint: .asciiz "\nHint: A planet with intelligent species!\n "

ColorHint: .asciiz "\nHint: Color associated with Death\n"

#Display

default1: .asciiz "\t\t\t\t\t |-----|\n"

default2: .asciiz "\t\t\t\t\t | |\n"

default3: .asciiz "\t\t\t\t\t\t|\n"

default4: .asciiz "\t\t\t\t\t\t|\n"

default5: .asciiz "\t\t\t\t\t\t|\n"

default6: .asciiz "\t\t\t\t\t\t|\n"

default7: .asciiz "\t\t\t\t\t\t|\n"

default8: .asciiz "\t\t\t\t\t---------\n"

first1: .asciiz "\t\t\t\t\t |-----| \n"

first2: .asciiz "\t\t\t\t\t | |\n"

first3: .asciiz "\t\t\t\t\t 0 |\n"

first4: .asciiz "\t\t\t\t\t\t|\n"

first5: .asciiz "\t\t\t\t\t\t|\n"

first6: .asciiz "\t\t\t\t\t\t|\n"

first7: .asciiz "\t\t\t\t\t\t|\n"

first8: .asciiz "\t\t\t\t\t---------\n"

first9: .asciiz "\nYou have 5 chances left"

second1: .asciiz "\t\t\t\t\t |-----| \n"

second2: .asciiz "\t\t\t\t\t | |\n"

second3: .asciiz "\t\t\t\t\t 0 |\n"

second4: .asciiz "\t\t\t\t\t | |\n"

second5: .asciiz "\t\t\t\t\t | |\n"

second6: .asciiz "\t\t\t\t\t\t|\n"

second7: .asciiz "\t\t\t\t\t\t|\n"

second8: .asciiz "\t\t\t\t\t---------\n"

second9: .asciiz "\nYou have 4 chances left"

third1: .asciiz "\t\t\t\t\t |-----| \n"

third2: .asciiz "\t\t\t\t\t | |\n"

third3: .asciiz "\t\t\t\t\t 0 |\n"

third4: .asciiz "\t\t\t\t\t \\| |\n"

third5: .asciiz "\t\t\t\t\t | |\n"

third6: .asciiz "\t\t\t\t\t\t|\n"

third7: .asciiz "\t\t\t\t\t\t|\n"

third8: .asciiz "\t\t\t\t\t---------\n"

third9: .asciiz "\nYou have 3 chances left"

fourth1: .asciiz "\t\t\t\t\t |-----| \n"

fourth2: .asciiz "\t\t\t\t\t | |\n"

fourth3: .asciiz "\t\t\t\t\t 0 |\n"

fourth4: .asciiz "\t\t\t\t\t \\|/ |\n"

fourth5: .asciiz "\t\t\t\t\t | |\n"

fourth6: .asciiz "\t\t\t\t\t\t|\n"

fourth7: .asciiz "\t\t\t\t\t\t|\n"

fourth8: .asciiz "\t\t\t\t\t---------\n"

fourth9: .asciiz "\nYou have 2 chances left"

fifth1: .asciiz "\t\t\t\t\t |-----| \n"

fifth2: .asciiz "\t\t\t\t\t | |\n"

fifth3: .asciiz "\t\t\t\t\t 0 |\n"

fifth4: .asciiz "\t\t\t\t\t \\|/ |\n"

fifth5: .asciiz "\t\t\t\t\t | |\n"

fifth6: .asciiz "\t\t\t\t\t / |\n"

fifth7: .asciiz "\t\t\t\t\t\t|\n"

fifth8: .asciiz "\t\t\t\t\t---------\n"

fifth9: .asciiz "\nYou have 1 chances left"

sixth1: .asciiz "\t\t\t\t\t |-----| \n"

sixth2: .asciiz "\t\t\t\t\t | |\n"

sixth3: .asciiz "\t\t\t\t\t 0 |\n"

sixth4: .asciiz "\t\t\t\t\t \\|/ |\n"

sixth5: .asciiz "\t\t\t\t\t | |\n"

sixth6: .asciiz "\t\t\t\t\t / \\ |\n"

sixth7: .asciiz "\t\t\t\t\t\t|\n"

sixth8: .asciiz "\t\t\t\t\t---------\n"

#input

input: .space 20

.text

.globl main

main:

j Countrycode

Countrycode:

#constants

li $t0, 0

li $t1, 0

li $t2, 0

li $t3, 0

li $t4, 0

li $s2, 0

li $t3, 0

li $t6, 0

li $t7, 0

li $t8, 0

li $t9, 0

li $s3, 0

addi $t4, $t4, 0 #miss counter

addi $s2,$s2,0#score

addi $t3,$t3,-1#counter variable for C

addi $t6,$t6,-1#counter variable for H

addi $t7,$t7,-1#counter variable for i

addi $t8,$t8,-1#counter variable for N

addi $t9,$t9,-1#counter variable for A

addi $s3,$s3,0#counter variable for All letters

#main menu

#output message

li $v0,4

la $a0,message1

syscall

#jump to the next code

b countletters

countletters:

#code to count letters in word

li $t1,0

la $t0,Country

#count loop

loop:

lb $a0,0($t0)

beqz $a0,done

addi $t0,$t0,1

addi $t1,$t1,1

j loop

#display for counted letters

done:

#display messsage

li $v0,4

la $a0,message2

syscall

#display counted letters

li $v0,1

add $a0, $0,$t1

syscall

b printdefault

printdefault:

#default print

default:

li $v0, 4

la $a0, default1

syscall

li $v0, 4

la $a0, default2

syscall

li $v0, 4

la $a0, default3

syscall

li $v0, 4

la $a0, default4

syscall

li $v0, 4

la $a0, default5

syscall

li $v0, 4

la $a0, default6

syscall

li $v0, 4

la $a0, default7

syscall

li $v0, 4

la $a0, default8

syscall

wincheck:

beq $s3,15,youwin

j checker

checker:

#display messsage to ask for guess

li $v0,4

la $a0,message7

syscall

# get input from user

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

#check all letters

li $s1, 'C' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalC # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'c' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalC # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'H' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalH # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'h' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalH # branch to "equal" label if $s0 and $s1 are

li $s1, 'I' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalI # branch to "equal" label if $s0 and $s1 are

li $s1, 'i' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalI # branch to "equal" label if $s0 and $s1 are

li $s1, 'N' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalN # branch to "equal" label if $s0 and $s1 are

li $s1, 'n' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalN # branch to "equal" label if $s0 and $s1 are

li $s1, 'A' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalA # branch to "equal" label if $s0 and $s1 are

li $s1, 'a' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalA # branch to "equal" label if $s0 and $s1 are

addi $t4,$t4,1

beq $t4,1,firstwrong

j nottrue1

nottrue1:

beq $t4,2,secondwrong

j nottrue2

nottrue2:

beq $t4,3,thirdwrong

j nottrue3

nottrue3:

beq $t4,4,fourthwrong

j nottrue4

nottrue4:

beq $t4,5,fifthwrong

j nottrue5

nottrue5:

beq $t4,6,sixthwrong

firstwrong:

li $v0,4

la $a0,message3

syscall

first:

li $v0, 4

la $a0, first1

syscall

li $v0, 4

la $a0, first2

syscall

li $v0, 4

la $a0, first3

syscall

li $v0, 4

la $a0, first4

syscall

li $v0, 4

la $a0, first5

syscall

li $v0, 4

la $a0, first6

syscall

li $v0, 4

la $a0, first7

syscall

li $v0, 4

la $a0, first8

syscall

li $v0, 4

la $a0, first9

syscall

j checker

secondwrong:

li $v0,4

la $a0,message3

syscall

li $v0, 4

la $a0, second1

syscall

li $v0, 4

la $a0, second2

syscall

li $v0, 4

la $a0, second3

syscall

li $v0, 4

la $a0, second4

syscall

li $v0, 4

la $a0, second5

syscall

li $v0, 4

la $a0, second6

syscall

li $v0, 4

la $a0, second7

syscall

li $v0, 4

la $a0, second8

syscall

li $v0, 4

la $a0, second9

syscall

j checker

thirdwrong:

li $v0,4

la $a0,message3

syscall

third:

li $v0, 4

la $a0, third1

syscall

li $v0, 4

la $a0, third2

syscall

li $v0, 4

la $a0, third3

syscall

li $v0, 4

la $a0, third4

syscall

li $v0, 4

la $a0, third5

syscall

li $v0, 4

la $a0, third6

syscall

li $v0, 4

la $a0, third7

syscall

li $v0, 4

la $a0, third8

syscall

li $v0, 4

la $a0, third9

syscall

j checker

fourthwrong:

li $v0,4

la $a0,message3

syscall

fourth:

li $v0, 4

la $a0, fourth1

syscall

li $v0, 4

la $a0, fourth2

syscall

li $v0, 4

la $a0, fourth3

syscall

li $v0, 4

la $a0, fourth4

syscall

li $v0, 4

la $a0, fourth5

syscall

li $v0, 4

la $a0, fourth6

syscall

li $v0, 4

la $a0, fourth7

syscall

li $v0, 4

la $a0, fourth8

syscall

li $v0, 4

la $a0, fourth9

syscall

li $v0, 4

la $a0, CountryHint

syscall

j checker

fifthwrong:

li $v0,4

la $a0,message3

syscall

fifth:

li $v0, 4

la $a0, fifth1

syscall

li $v0, 4

la $a0, fifth2

syscall

li $v0, 4

la $a0, fifth3

syscall

li $v0, 4

la $a0, fifth4

syscall

li $v0, 4

la $a0, fifth5

syscall

li $v0, 4

la $a0, fifth6

syscall

li $v0, 4

la $a0, fifth7

syscall

li $v0, 4

la $a0, fifth8

syscall

li $v0, 4

la $a0, fifth9

syscall

j checker

sixthwrong:

li $v0,4

la $a0,message3

syscall

sixth:

li $v0, 4

la $a0, sixth1

syscall

li $v0, 4

la $a0, sixth2

syscall

li $v0, 4

la $a0, sixth3

syscall

li $v0, 4

la $a0, sixth4

syscall

li $v0, 4

la $a0, sixth5

syscall

li $v0, 4

la $a0, sixth6

syscall

li $v0, 4

la $a0, sixth7

syscall

li $v0, 4

la $a0, sixth8

syscall

j gameover

equalC:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t3,$t3,1

bnez $t3,dontadd

j addvalueC

addvalueC:

addi $s3,$s3,1

j wincheck

dontadd:

j wincheck

equalH:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t6,$t6,1

bnez $t6,dontadd

j addvalueH

addvalueH:

addi $s3,$s3,2

j wincheck

equalI:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t7,$t7,1

bnez $t7,dontadd

j addvalueI

addvalueI:

addi $s3,$s3,3

j wincheck

equalN:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t8,$t8,1

bnez $t8,dontadd

j addvalueN

addvalueN:

addi $s3,$s3,4

j wincheck

equalA:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t9,$t9,1

bnez $t9,dontadd

j addvalueA

addvalueA:

addi $s3,$s3,5

j wincheck

gameover:

li $v0,4

la $a0,message5

syscall

li $v0,4

la $a0,message8

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Countrycode # branch to "equal" label if $s0 and $s1 are equal

j exit

youwin:

li $v0,4

la $a0,message6

syscall

li $v0,4

la $a0,message9

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Animalcode # branch to "equal" label if $s0 and $s1 are equal

exit:

li $v0, 10 # system call for exit

syscall

Animalcode:

#constants

li $s0,0

li $s1,0

li $s2, 0

li $s3, 0

li $s4, 0

li $s5, 0

li $s6, 0

li $s7, 0

li $t0, 0

li $t1, 0

li $t2, 0

li $t3, 0

li $t4, 0

li $t3, 0

li $t6, 0

li $t7, 0

li $t8, 0

li $t9, 0

addi $t4, $t4, 0 #miss counter

addi $s2,$s2,0#score

addi $t3,$t3,-1#counter variable for H

addi $t6,$t6,-1#counter variable for Y

addi $t7,$t7,-1#counter variable for E

addi $t8,$t8,-1#counter variable for N

addi $t9,$t9,-1#counter variable for A

addi $s3,$s3,0#counter variable for All letters

#main menu

#output message

#jump to the next code

b countlettersW

countlettersW:

#code to count letters in word

li $t1,0

la $t0,Animal

#count loop

loopW:

lb $a0,0($t0)

beqz $a0,doneW

addi $t0,$t0,1

addi $t1,$t1,1

j loopW

#display for counted letters

doneW:

#display messsage

li $v0,4

la $a0,message2

syscall

#display counted letters

li $v0,1

add $a0, $0,$t1

syscall

b printdefaultW

printdefaultW:

#default print

defaultW:

li $v0, 4

la $a0, default1

syscall

li $v0, 4

la $a0, default2

syscall

li $v0, 4

la $a0, default3

syscall

li $v0, 4

la $a0, default4

syscall

li $v0, 4

la $a0, default5

syscall

li $v0, 4

la $a0, default6

syscall

li $v0, 4

la $a0, default7

syscall

li $v0, 4

la $a0, default8

syscall

wincheckW:

beq $s3,15,youwinW

j checkerW

checkerW:

#display messsage to ask for guess

li $v0,4

la $a0,message7

syscall

# get input from user

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

#check all letters

li $s1, 'H' # load ASCII value of 'H' into register $s1

beq $s0, $s1, equalHW # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'h' # load ASCII value of 'H' into register $s1

beq $s0, $s1, equalHW # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'Y' # load ASCII value of 'Y' into register $s1

beq $s0, $s1, equalYW # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'y' # load ASCII value of 'Y' into register $s1

beq $s0, $s1, equalYW # branch to "equal" label if $s0 and $s1 are

li $s1, 'E' # load ASCII value of 'E' into register $s1

beq $s0, $s1, equalEW # branch to "equal" label if $s0 and $s1 are

li $s1, 'e' # load ASCII value of 'E' into register $s1

beq $s0, $s1, equalEW # branch to "equal" label if $s0 and $s1 are

li $s1, 'N' # load ASCII value of 'N' into register $s1

beq $s0, $s1, equalNW # branch to "equal" label if $s0 and $s1 are

li $s1, 'n' # load ASCII value of 'N' into register $s1

beq $s0, $s1, equalNW # branch to "equal" label if $s0 and $s1 are

li $s1, 'A' # load ASCII value of 'A' into register $s1

beq $s0, $s1, equalAW # branch to "equal" label if $s0 and $s1 are

li $s1, 'a' # load ASCII value of 'A' into register $s1

beq $s0, $s1, equalAW # branch to "equal" label if $s0 and $s1 are

addi $t4,$t4,1

beq $t4,1,firstwrongW

j nottrue1W

nottrue1W:

beq $t4,2,secondwrongW

j nottrue2W

nottrue2W:

beq $t4,3,thirdwrongW

j nottrue3W

nottrue3W:

beq $t4,4,fourthwrongW

j nottrue4W

nottrue4W:

beq $t4,5,fifthwrongW

j nottrue5W

nottrue5W:

beq $t4,6,sixthwrongW

firstwrongW:

li $v0,4

la $a0,message3

syscall

firstW:

li $v0, 4

la $a0, first1

syscall

li $v0, 4

la $a0, first2

syscall

li $v0, 4

la $a0, first3

syscall

li $v0, 4

la $a0, first4

syscall

li $v0, 4

la $a0, first5

syscall

li $v0, 4

la $a0, first6

syscall

li $v0, 4

la $a0, first7

syscall

li $v0, 4

la $a0, first8

syscall

li $v0, 4

la $a0, first9

syscall

j checkerW

secondwrongW:

li $v0,4

la $a0,message3

syscall

li $v0, 4

la $a0, second1

syscall

li $v0, 4

la $a0, second2

syscall

li $v0, 4

la $a0, second3

syscall

li $v0, 4

la $a0, second4

syscall

li $v0, 4

la $a0, second5

syscall

li $v0, 4

la $a0, second6

syscall

li $v0, 4

la $a0, second7

syscall

li $v0, 4

la $a0, second8

syscall

li $v0, 4

la $a0, second9

syscall

j checkerW

thirdwrongW:

li $v0,4

la $a0,message3

syscall

thirdW:

li $v0, 4

la $a0, third1

syscall

li $v0, 4

la $a0, third2

syscall

li $v0, 4

la $a0, third3

syscall

li $v0, 4

la $a0, third4

syscall

li $v0, 4

la $a0, third5

syscall

li $v0, 4

la $a0, third6

syscall

li $v0, 4

la $a0, third7

syscall

li $v0, 4

la $a0, third8

syscall

li $v0, 4

la $a0, third9

syscall

j checkerW

fourthwrongW:

li $v0,4

la $a0,message3

syscall

fourthW:

li $v0, 4

la $a0, fourth1

syscall

li $v0, 4

la $a0, fourth2

syscall

li $v0, 4

la $a0, fourth3

syscall

li $v0, 4

la $a0, fourth4

syscall

li $v0, 4

la $a0, fourth5

syscall

li $v0, 4

la $a0, fourth6

syscall

li $v0, 4

la $a0, fourth7

syscall

li $v0, 4

la $a0, fourth8

syscall

li $v0, 4

la $a0, fourth9

syscall

li $v0, 4

la $a0, AnimalHint

syscall

j checkerW

fifthwrongW:

li $v0,4

la $a0,message3

syscall

fifthW:

li $v0, 4

la $a0, fifth1

syscall

li $v0, 4

la $a0, fifth2

syscall

li $v0, 4

la $a0, fifth3

syscall

li $v0, 4

la $a0, fifth4

syscall

li $v0, 4

la $a0, fifth5

syscall

li $v0, 4

la $a0, fifth6

syscall

li $v0, 4

la $a0, fifth7

syscall

li $v0, 4

la $a0, fifth8

syscall

li $v0, 4

la $a0, fifth9

syscall

j checkerW

sixthwrongW:

li $v0,4

la $a0,message3

syscall

sixthW:

li $v0, 4

la $a0, sixth1

syscall

li $v0, 4

la $a0, sixth2

syscall

li $v0, 4

la $a0, sixth3

syscall

li $v0, 4

la $a0, sixth4

syscall

li $v0, 4

la $a0, sixth5

syscall

li $v0, 4

la $a0, sixth6

syscall

li $v0, 4

la $a0, sixth7

syscall

li $v0, 4

la $a0, sixth8

syscall

j gameoverW

gameoverW:

li $v0,4

la $a0,message5

syscall

li $v0,4

la $a0,message8

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Animalcode # branch to "equal" label if $s0 and $s1 are equal

j exit

equalHW:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t3,$t3,1

bnez $t3,dontaddW

j addvalueHW

addvalueHW:

addi $s3,$s3,1

j wincheckW

dontaddW:

j wincheckW

equalYW:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t6,$t6,1

bnez $t6,dontaddW

j addvalueYW

addvalueYW:

addi $s3,$s3,2

j wincheckW

equalEW:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t7,$t7,1

bnez $t7,dontaddW

j addvalueEW

addvalueEW:

addi $s3,$s3,3

j wincheckW

equalNW:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t8,$t8,1

bnez $t8,dontaddW

j addvalueNW

addvalueNW:

addi $s3,$s3,4

j wincheckW

equalAW:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t9,$t9,1

bnez $t9,dontaddW

j addvalueAW

addvalueAW:

addi $s3,$s3,5

j wincheckW

youwinW:

li $v0,4

la $a0,message10

syscall

li $v0,4

la $a0,message9

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Yachtcode # branch to "equal" label if $s0 and $s1 are equal

j exit

exitW:

li $v0, 10 # system call for exit

syscall

Yachtcode:

#constants

li $t0 , 0

li $t1 , 0

li $t2 , 0

li $t4 , 0

li $s2 , 0

li $t3 , 0

li $t6 , 0

li $t7 , 0

li $t8 , 0

li $t9 , 0

li $s3 , 0

addi $t4, $t4, 0 #miss counter

addi $s2,$s2,0#score

addi $t3,$t3,-1#counter variable for C

addi $t6,$t6,-1#counter variable for H

addi $t7,$t7,-1#counter variable for i

addi $t8,$t8,-1#counter variable for N

addi $t9,$t9,-1#counter variable for A

addi $s3,$s3,0#counter variable for All letters

#jump to the next code

b countlettersX

countlettersX:

#code to count letters in word

li $t1,0

la $t0,Yacht

#count loop

loopX:

lb $a0,0($t0)

beqz $a0,doneX

addi $t0,$t0,1

addi $t1,$t1,1

j loopX

#display for counted letters

doneX:

#display messsage

li $v0,4

la $a0,message2

syscall

#display counted letters

li $v0,1

add $a0, $0,$t1

syscall

b printdefaultX

printdefaultX:

#default print

defaultX:

li $v0, 4

la $a0, default1

syscall

li $v0, 4

la $a0, default2

syscall

li $v0, 4

la $a0, default3

syscall

li $v0, 4

la $a0, default4

syscall

li $v0, 4

la $a0, default5

syscall

li $v0, 4

la $a0, default6

syscall

li $v0, 4

la $a0, default7

syscall

li $v0, 4

la $a0, default8

syscall

wincheckX:

beq $s3,15,youwinX

j checkerX

checkerX:

#display messsage to ask for guess

li $v0,4

la $a0,message7

syscall

# get input from user

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

#check all letters

li $s1, 'Y' # load ASCII value of 'Y' into register $s1

beq $s0, $s1, equalYX # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'y' # load ASCII value of 'y' into register $s1

beq $s0, $s1, equalYX# branch to "equal" label if $s0 and $s1 are equal

li $s1, 'A' # load ASCII value of 'A' into register $s1

beq $s0, $s1, equalAX # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'a' # load ASCII value of 'a' into register $s1

beq $s0, $s1, equalAX # branch to "equal" label if $s0 and $s1 are

li $s1, 'C' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalCX # branch to "equal" label if $s0 and $s1 are

li $s1, 'c' # load ASCII value of 'c' into register $s1

beq $s0, $s1, equalCX # branch to "equal" label if $s0 and $s1 are

li $s1, 'H' # load ASCII value of 'H' into register $s1

beq $s0, $s1, equalHX # branch to "equal" label if $s0 and $s1 are

li $s1, 'h' # load ASCII value of 'h' into register $s1

beq $s0, $s1, equalHX # branch to "equal" label if $s0 and $s1 are

li $s1, 'T' # load ASCII value of 'T' into register $s1

beq $s0, $s1, equalTX # branch to "equal" label if $s0 and $s1 are

li $s1, 't' # load ASCII value of 't' into register $s1

beq $s0, $s1, equalTX # branch to "equal" label if $s0 and $s1 are

addi $t4,$t4,1

beq $t4,1,firstwrongX

j nottrue1X

nottrue1X:

beq $t4,2,secondwrongX

j nottrue2X

nottrue2X:

beq $t4,3,thirdwrongX

j nottrue3X

nottrue3X:

beq $t4,4,fourthwrongX

j nottrue4X

nottrue4X:

beq $t4,5,fifthwrongX

j nottrue5X

nottrue5X:

beq $t4,6,sixthwrongX

firstwrongX:

li $v0,4

la $a0,message3

syscall

firstX:

li $v0, 4

la $a0, first1

syscall

li $v0, 4

la $a0, first2

syscall

li $v0, 4

la $a0, first3

syscall

li $v0, 4

la $a0, first4

syscall

li $v0, 4

la $a0, first5

syscall

li $v0, 4

la $a0, first6

syscall

li $v0, 4

la $a0, first7

syscall

li $v0, 4

la $a0, first8

syscall

li $v0, 4

la $a0, first9

syscall

j checkerX

secondwrongX:

li $v0,4

la $a0,message3

syscall

li $v0, 4

la $a0, second1

syscall

li $v0, 4

la $a0, second2

syscall

li $v0, 4

la $a0, second3

syscall

li $v0, 4

la $a0, second4

syscall

li $v0, 4

la $a0, second5

syscall

li $v0, 4

la $a0, second6

syscall

li $v0, 4

la $a0, second7

syscall

li $v0, 4

la $a0, second8

syscall

li $v0, 4

la $a0, second9

syscall

j checkerX

thirdwrongX:

li $v0,4

la $a0,message3

syscall

thirdX:

li $v0, 4

la $a0, third1

syscall

li $v0, 4

la $a0, third2

syscall

li $v0, 4

la $a0, third3

syscall

li $v0, 4

la $a0, third4

syscall

li $v0, 4

la $a0, third5

syscall

li $v0, 4

la $a0, third6

syscall

li $v0, 4

la $a0, third7

syscall

li $v0, 4

la $a0, third8

syscall

li $v0, 4

la $a0, third9

syscall

j checkerX

fourthwrongX:

li $v0,4

la $a0,message3

syscall

fourthX:

li $v0, 4

la $a0, fourth1

syscall

li $v0, 4

la $a0, fourth2

syscall

li $v0, 4

la $a0, fourth3

syscall

li $v0, 4

la $a0, fourth4

syscall

li $v0, 4

la $a0, fourth5

syscall

li $v0, 4

la $a0, fourth6

syscall

li $v0, 4

la $a0, fourth7

syscall

li $v0, 4

la $a0, fourth8

syscall

li $v0, 4

la $a0, fourth9

syscall

li $v0, 4

la $a0, YachtHint

syscall

j checkerX

fifthwrongX:

li $v0,4

la $a0,message3

syscall

fifthX:

li $v0, 4

la $a0, fifth1

syscall

li $v0, 4

la $a0, fifth2

syscall

li $v0, 4

la $a0, fifth3

syscall

li $v0, 4

la $a0, fifth4

syscall

li $v0, 4

la $a0, fifth5

syscall

li $v0, 4

la $a0, fifth6

syscall

li $v0, 4

la $a0, fifth7

syscall

li $v0, 4

la $a0, fifth8

syscall

li $v0, 4

la $a0, fifth9

syscall

j checkerX

sixthwrongX:

li $v0,4

la $a0,message3

syscall

sixthX:

li $v0, 4

la $a0, sixth1

syscall

li $v0, 4

la $a0, sixth2

syscall

li $v0, 4

la $a0, sixth3

syscall

li $v0, 4

la $a0, sixth4

syscall

li $v0, 4

la $a0, sixth5

syscall

li $v0, 4

la $a0, sixth6

syscall

li $v0, 4

la $a0, sixth7

syscall

li $v0, 4

la $a0, sixth8

syscall

j gameoverX

gameoverX:

li $v0,4

la $a0,message5

syscall

li $v0,4

la $a0,message8

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Yachtcode # branch to "equal" label if $s0 and $s1 are equal

j exit

equalYX:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t3,$t3,1

bnez $t3,dontaddX

j addvalueYX

addvalueYX:

addi $s3,$s3,1

j wincheckX

dontaddX:

j wincheckX

equalAX:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t6,$t6,1

bnez $t6,dontaddX

j addvalueAX

addvalueAX:

addi $s3,$s3,2

j wincheckX

equalCX:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t7,$t7,1

bnez $t7,dontaddX

j addvalueCX

addvalueCX:

addi $s3,$s3,3

j wincheckX

equalTX:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t8,$t8,1

bnez $t8,dontaddX

j addvalueTX

addvalueTX:

addi $s3,$s3,4

j wincheckX

equalHX:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t9,$t9,1

bnez $t9,dontaddX

j addvalueHX

addvalueHX:

addi $s3,$s3,5

j wincheckX

youwinX:

li $v0,4

la $a0,message11

syscall

li $v0,4

la $a0,message9

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Earthcode # branch to "equal" label if $s0 and $s1 are equal

j exit

exitX:

li $v0, 10 # system call for exit

syscall

Earthcode:

#constants

li $t0 , 0

li $t1 , 0

li $t2 , 0

li $t4 , 0

li $s2 , 0

li $t3 , 0

li $t6 , 0

li $t7 , 0

li $t8 , 0

li $t9 , 0

li $s3 , 0

addi $t4, $t4, 0 #miss counter

addi $s2,$s2,0#score

addi $t3,$t3,-1#counter variable for C

addi $t6,$t6,-1#counter variable for H

addi $t7,$t7,-1#counter variable for i

addi $t8,$t8,-1#counter variable for N

addi $t9,$t9,-1#counter variable for A

addi $s3,$s3,0#counter variable for All letters

#jump to the next code

b countlettersY

countlettersY:

#code to count letters in word

li $t1,0

la $t0,Earth

#count loop

loopY:

lb $a0,0($t0)

beqz $a0,doneY

addi $t0,$t0,1

addi $t1,$t1,1

j loopY

#display for counted letters

doneY:

#display messsage

li $v0,4

la $a0,message2

syscall

#display counted letters

li $v0,1

add $a0, $0,$t1

syscall

b printdefaultY

printdefaultY:

#default print

defaultY:

li $v0, 4

la $a0, default1

syscall

li $v0, 4

la $a0, default2

syscall

li $v0, 4

la $a0, default3

syscall

li $v0, 4

la $a0, default4

syscall

li $v0, 4

la $a0, default5

syscall

li $v0, 4

la $a0, default6

syscall

li $v0, 4

la $a0, default7

syscall

li $v0, 4

la $a0, default8

syscall

wincheckY:

beq $s3,15,youwinY

j checkerY

checkerY:

#display messsage to ask for guess

li $v0,4

la $a0,message7

syscall

# get input from user

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

#check all letters

li $s1, 'E' # load ASCII value of 'E' into register $s1

beq $s0, $s1, equalEY # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'e' # load ASCII value of 'e' into register $s1

beq $s0, $s1, equalEY# branch to "equal" label if $s0 and $s1 are equal

li $s1, 'A' # load ASCII value of 'A' into register $s1

beq $s0, $s1, equalAY # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'a' # load ASCII value of 'a' into register $s1

beq $s0, $s1, equalAY # branch to "equal" label if $s0 and $s1 are

li $s1, 'R' # load ASCII value of 'R' into register $s1

beq $s0, $s1, equalRY # branch to "equal" label if $s0 and $s1 are

li $s1, 'r' # load ASCII value of 'r' into register $s1

beq $s0, $s1, equalRY # branch to "equal" label if $s0 and $s1 are

li $s1, 'T' # load ASCII value of 'T' into register $s1

beq $s0, $s1, equalTY # branch to "equal" label if $s0 and $s1 are

li $s1, 't' # load ASCII value of 't' into register $s1

beq $s0, $s1, equalTY # branch to "equal" label if $s0 and $s1 are

li $s1, 'H' # load ASCII value of 'T' into register $s1

beq $s0, $s1, equalHY # branch to "equal" label if $s0 and $s1 are

li $s1, 'h' # load ASCII value of 't' into register $s1

beq $s0, $s1, equalHY # branch to "equal" label if $s0 and $s1 are

addi $t4,$t4,1

beq $t4,1,firstwrongY

j nottrue1Y

nottrue1Y:

beq $t4,2,secondwrongY

j nottrue2Y

nottrue2Y:

beq $t4,3,thirdwrongY

j nottrue3Y

nottrue3Y:

beq $t4,4,fourthwrongY

j nottrue4Y

nottrue4Y:

beq $t4,5,fifthwrongY

j nottrue5Y

nottrue5Y:

beq $t4,6,sixthwrongY

firstwrongY:

li $v0,4

la $a0,message3

syscall

firstY:

li $v0, 4

la $a0, first1

syscall

li $v0, 4

la $a0, first2

syscall

li $v0, 4

la $a0, first3

syscall

li $v0, 4

la $a0, first4

syscall

li $v0, 4

la $a0, first5

syscall

li $v0, 4

la $a0, first6

syscall

li $v0, 4

la $a0, first7

syscall

li $v0, 4

la $a0, first8

syscall

li $v0, 4

la $a0, first9

syscall

j checkerY

secondwrongY:

li $v0,4

la $a0,message3

syscall

li $v0, 4

la $a0, second1

syscall

li $v0, 4

la $a0, second2

syscall

li $v0, 4

la $a0, second3

syscall

li $v0, 4

la $a0, second4

syscall

li $v0, 4

la $a0, second5

syscall

li $v0, 4

la $a0, second6

syscall

li $v0, 4

la $a0, second7

syscall

li $v0, 4

la $a0, second8

syscall

li $v0, 4

la $a0, second9

syscall

j checkerY

thirdwrongY:

li $v0,4

la $a0,message3

syscall

thirdY:

li $v0, 4

la $a0, third1

syscall

li $v0, 4

la $a0, third2

syscall

li $v0, 4

la $a0, third3

syscall

li $v0, 4

la $a0, third4

syscall

li $v0, 4

la $a0, third5

syscall

li $v0, 4

la $a0, third6

syscall

li $v0, 4

la $a0, third7

syscall

li $v0, 4

la $a0, third8

syscall

li $v0, 4

la $a0, third9

syscall

j checkerY

fourthwrongY:

li $v0,4

la $a0,message3

syscall

fourthY:

li $v0, 4

la $a0, fourth1

syscall

li $v0, 4

la $a0, fourth2

syscall

li $v0, 4

la $a0, fourth3

syscall

li $v0, 4

la $a0, fourth4

syscall

li $v0, 4

la $a0, fourth5

syscall

li $v0, 4

la $a0, fourth6

syscall

li $v0, 4

la $a0, fourth7

syscall

li $v0, 4

la $a0, fourth8

syscall

li $v0, 4

la $a0, fourth9

syscall

li $v0, 4

la $a0, EarthHint

syscall

j checkerY

fifthwrongY:

li $v0,4

la $a0,message3

syscall

fifthY:

li $v0, 4

la $a0, fifth1

syscall

li $v0, 4

la $a0, fifth2

syscall

li $v0, 4

la $a0, fifth3

syscall

li $v0, 4

la $a0, fifth4

syscall

li $v0, 4

la $a0, fifth5

syscall

li $v0, 4

la $a0, fifth6

syscall

li $v0, 4

la $a0, fifth7

syscall

li $v0, 4

la $a0, fifth8

syscall

li $v0, 4

la $a0, fifth9

syscall

j checkerY

sixthwrongY:

li $v0,4

la $a0,message3

syscall

sixthY: #Hangman Display for 6th association

li $v0, 4

la $a0, sixth1

syscall

li $v0, 4

la $a0, sixth2

syscall

li $v0, 4

la $a0, sixth3

syscall

li $v0, 4

la $a0, sixth4

syscall

li $v0, 4

la $a0, sixth5

syscall

li $v0, 4

la $a0, sixth6

syscall

li $v0, 4

la $a0, sixth7

syscall

li $v0, 4

la $a0, sixth8

syscall

j gameoverY

gameoverY:

li $v0,4

la $a0,message5

syscall

li $v0,4

la $a0,message8

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Earthcode # branch to "equal" label if $s0 and $s1 are equal

j exit

equalEY:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t3,$t3,1

bnez $t3,dontaddY

j addvalueEY

addvalueEY:

addi $s3,$s3,1

j wincheckY

dontaddY:

j wincheckY

equalAY:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t6,$t6,1

bnez $t6,dontaddY

j addvalueAY

addvalueAY:

addi $s3,$s3,2

j wincheckY

equalRY:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t7,$t7,1

bnez $t7,dontaddY

j addvalueRY

addvalueRY:

addi $s3,$s3,3

j wincheckY

equalTY:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t8,$t8,1

bnez $t8,dontaddY

j addvalueTY

addvalueTY:

addi $s3,$s3,4

j wincheckY

equalHY:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t9,$t9,1

bnez $t9,dontaddY

j addvalueHY

addvalueHY:

addi $s3,$s3,5

j wincheckY

youwinY:

li $v0,4

la $a0,message12

syscall

li $v0,4

la $a0,message9

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Colorcode # branch to "equal" label if $s0 and $s1 are equal

j exit

exitY:

li $v0, 10 # system call for exit

syscall

Colorcode:

#constants

li $t0, 0

li $t1, 0

li $t2, 0

li $t3, 0

li $t4, 0

li $s2, 0

li $t3, 0

li $t6, 0

li $t7, 0

li $t8, 0

li $t9, 0

li $s3, 0

addi $t4, $t4, 0 #miss counter

addi $s2,$s2,0#score

addi $t3,$t3,-1#counter variable for B

addi $t6,$t6,-1#counter variable for L

addi $t7,$t7,-1#counter variable for A

addi $t8,$t8,-1#counter variable for C

addi $t9,$t9,-1#counter variable for K

addi $s3,$s3,0#counter variable for All letters

#main menu

#output message

#jump to the next code

b countlettersZ

countlettersZ:

#code to count letters in word

li $t1,0

la $t0,Color

#count loop

loopZ:

lb $a0,0($t0)

beqz $a0,doneZ

addi $t0,$t0,1

addi $t1,$t1,1

j loopZ

#display for counted letters

doneZ:

#display messsage

li $v0,4

la $a0,message2

syscall

#display counted letters

li $v0,1

add $a0, $0,$t1

syscall

b printdefaultZ

printdefaultZ:

#default print

defaultZ:

li $v0, 4

la $a0, default1

syscall

li $v0, 4

la $a0, default2

syscall

li $v0, 4

la $a0, default3

syscall

li $v0, 4

la $a0, default4

syscall

li $v0, 4

la $a0, default5

syscall

li $v0, 4

la $a0, default6

syscall

li $v0, 4

la $a0, default7

syscall

li $v0, 4

la $a0, default8

syscall

wincheckZ:

beq $s3,15,youwinZ

j checkerZ

checkerZ:

#display messsage to ask for guess

li $v0,4

la $a0,message7

syscall

# get input from user

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

#check all letters

li $s1, 'B' # load ASCII value of 'B' into register $s1

beq $s0, $s1, equalBZ # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'b' # load ASCII value of 'B' into register $s1

beq $s0, $s1, equalBZ # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'L' # load ASCII value of 'L' into register $s1

beq $s0, $s1, equalLZ # branch to "equal" label if $s0 and $s1 are equal

li $s1, 'l' # load ASCII value of 'L' into register $s1

beq $s0, $s1, equalLZ # branch to "equal" label if $s0 and $s1 are

li $s1, 'A' # load ASCII value of 'A' into register $s1

beq $s0, $s1, equalAZ # branch to "equal" label if $s0 and $s1 are

li $s1, 'a' # load ASCII value of 'A' into register $s1

beq $s0, $s1, equalAZ # branch to "equal" label if $s0 and $s1 are

li $s1, 'C' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalCZ # branch to "equal" label if $s0 and $s1 are

li $s1, 'c' # load ASCII value of 'C' into register $s1

beq $s0, $s1, equalCZ # branch to "equal" label if $s0 and $s1 are

li $s1, 'K' # load ASCII value of 'K' into register $s1

beq $s0, $s1, equalKZ # branch to "equal" label if $s0 and $s1 are

li $s1, 'k' # load ASCII value of 'K' into register $s1

beq $s0, $s1, equalKZ # branch to "equal" label if $s0 and $s1 are

addi $t4,$t4,1

beq $t4,1,firstwrongZ

j nottrue1Z

nottrue1Z:

beq $t4,2,secondwrongZ

j nottrue2Z

nottrue2Z:

beq $t4,3,thirdwrongZ

j nottrue3Z

nottrue3Z:

beq $t4,4,fourthwrongZ

j nottrue4Z

nottrue4Z:

beq $t4,5,fifthwrongZ

j nottrue5Z

nottrue5Z:

beq $t4,6,sixthwrongZ

firstwrongZ:

li $v0,4

la $a0,message3

syscall

firstZ:

li $v0, 4

la $a0, first1

syscall

li $v0, 4

la $a0, first2

syscall

li $v0, 4

la $a0, first3

syscall

li $v0, 4

la $a0, first4

syscall

li $v0, 4

la $a0, first5

syscall

li $v0, 4

la $a0, first6

syscall

li $v0, 4

la $a0, first7

syscall

li $v0, 4

la $a0, first8

syscall

li $v0, 4

la $a0, first9

syscall

j checkerZ

secondwrongZ:

li $v0,4

la $a0,message3

syscall

li $v0, 4

la $a0, second1

syscall

li $v0, 4

la $a0, second2

syscall

li $v0, 4

la $a0, second3

syscall

li $v0, 4

la $a0, second4

syscall

li $v0, 4

la $a0, second5

syscall

li $v0, 4

la $a0, second6

syscall

li $v0, 4

la $a0, second7

syscall

li $v0, 4

la $a0, second8

syscall

li $v0, 4

la $a0, second9

syscall

j checkerZ

thirdwrongZ:

li $v0,4

la $a0,message3

syscall

thirdZ:

li $v0, 4

la $a0, third1

syscall

li $v0, 4

la $a0, third2

syscall

li $v0, 4

la $a0, third3

syscall

li $v0, 4

la $a0, third4

syscall

li $v0, 4

la $a0, third5

syscall

li $v0, 4

la $a0, third6

syscall

li $v0, 4

la $a0, third7

syscall

li $v0, 4

la $a0, third8

syscall

li $v0, 4

la $a0, third9

syscall

j checkerZ

fourthwrongZ:

li $v0,4

la $a0,message3

syscall

fourthZ:

li $v0, 4

la $a0, fourth1

syscall

li $v0, 4

la $a0, fourth2

syscall

li $v0, 4

la $a0, fourth3

syscall

li $v0, 4

la $a0, fourth4

syscall

li $v0, 4

la $a0, fourth5

syscall

li $v0, 4

la $a0, fourth6

syscall

li $v0, 4

la $a0, fourth7

syscall

li $v0, 4

la $a0, fourth8

syscall

li $v0, 4

la $a0, fourth9

syscall

li $v0, 4

la $a0, ColorHint

syscall

j checkerZ

fifthwrongZ:

li $v0,4

la $a0,message3

syscall

fifthZ:

li $v0, 4

la $a0, fifth1

syscall

li $v0, 4

la $a0, fifth2

syscall

li $v0, 4

la $a0, fifth3

syscall

li $v0, 4

la $a0, fifth4

syscall

li $v0, 4

la $a0, fifth5

syscall

li $v0, 4

la $a0, fifth6

syscall

li $v0, 4

la $a0, fifth7

syscall

li $v0, 4

la $a0, fifth8

syscall

li $v0, 4

la $a0, fifth9

syscall

j checkerZ

sixthwrongZ:

li $v0,4

la $a0,message3

syscall

sixthZ:

li $v0, 4

la $a0, sixth1

syscall

li $v0, 4

la $a0, sixth2

syscall

li $v0, 4

la $a0, sixth3

syscall

li $v0, 4

la $a0, sixth4

syscall

li $v0, 4

la $a0, sixth5

syscall

li $v0, 4

la $a0, sixth6

syscall

li $v0, 4

la $a0, sixth7

syscall

li $v0, 4

la $a0, sixth8

syscall

j gameoverZ

gameoverZ:

li $v0,4

la $a0,message5

syscall

li $v0,4

la $a0,message8

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Colorcode # branch to "equal" label if $s0 and $s1 are equal

j exit

equalBZ:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t3,$t3,1

bnez $t3,dontaddZ

j addvalueBZ

addvalueBZ:

addi $s3,$s3,1

j wincheckZ

dontaddZ:

j wincheckZ

equalLZ:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t6,$t6,1

bnez $t6,dontaddZ

j addvalueLZ

addvalueLZ:

addi $s3,$s3,2

j wincheckZ

equalAZ:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t7,$t7,1

bnez $t7,dontaddZ

j addvalueAZ

addvalueAZ:

addi $s3,$s3,3

j wincheckZ

equalCZ:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t8,$t8,1

bnez $t8,dontaddZ

j addvalueCZ

addvalueCZ:

addi $s3,$s3,4

j wincheckZ

equalKZ:

#correct message

li $v0,4

la $a0,message4

syscall

addi $t9,$t9,1

bnez $t9,dontaddZ

j addvalueKZ

addvalueKZ:

addi $s3,$s3,5

j wincheckZ

youwinZ:

li $v0,4

la $a0,message13

syscall

li $v0,4

la $a0,message8

syscall

#record what user wants to do

li $v0, 12 # system call for reading a character

syscall

move $s0, $v0 # move input to $s0

li $s1, 'y' # load ASCII value of 'C' into register $s1

beq $s0, $s1,Countrycode # branch to "equal" label if $s0 and $s1 are equal

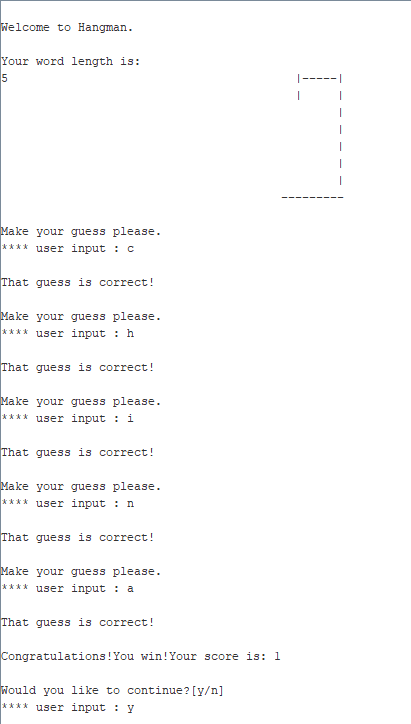
j exit

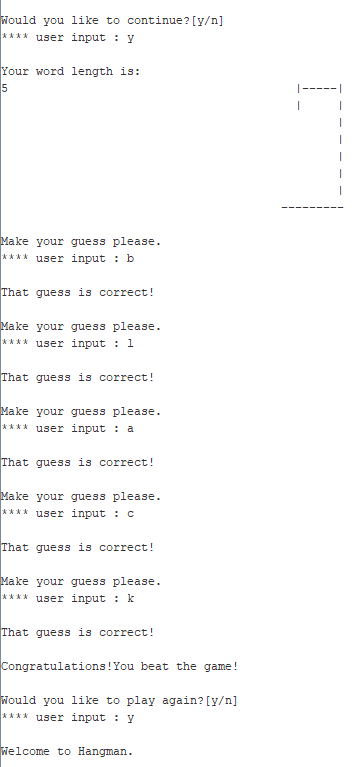
exitZ:

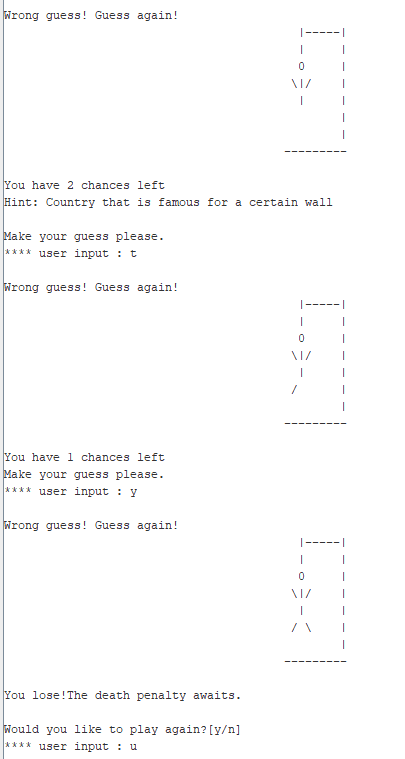
li $v0, 10 # system call for exit

syscall

1. INTERFACES







1. CONCLUSION

Through creating this project, we had a lot to learn about assembly language and how you can create a complex game from a set of registers and minimal commands. We also got to experience how to create a game from scratch on the MARS MIPS IDE Console. It made us realize how projects can turn out to be fun too once everyone cooperates. The project gave us an opportunity to enhance our group working skills as well and our MIPS coding skills.