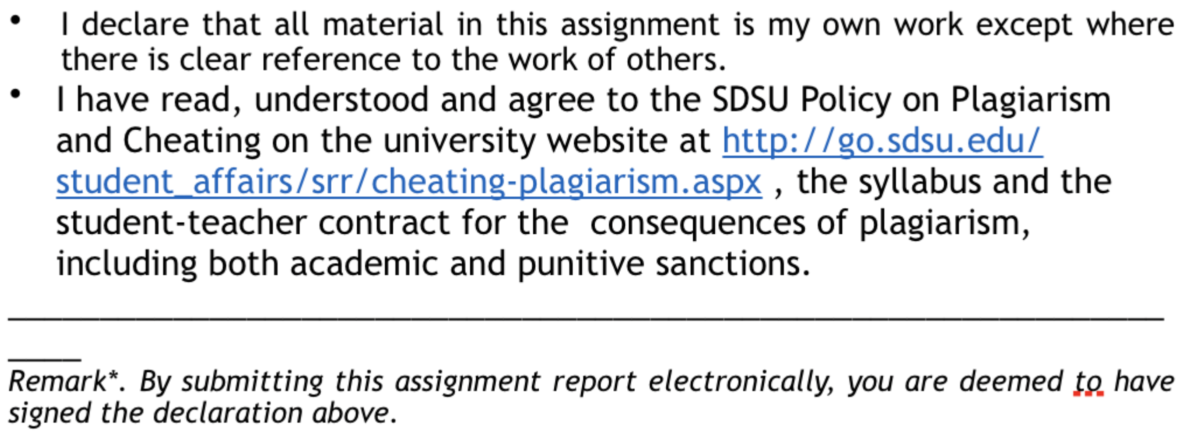
FINAL PROJECT REPORT

A Game of Blackcraps

AJ Rodrigo

04.08.2021

RedID: 821538725



- I declare that all material in this assignment is my own work except where there is clear reference to the work of others.
- I have read, understood and agree to the SDSU Policy on Plagiarism and Cheating on the university website at http://go.sdsu.edu/student_affairs/srr/cheating-plagiarism.aspx , the syllabus and the student-teacher contract for the consequences of plagiarism, including both academic and punitive sanctions.

Remark. By submitting this assignment report electronically, you are deemed to have signed the declaration above.*

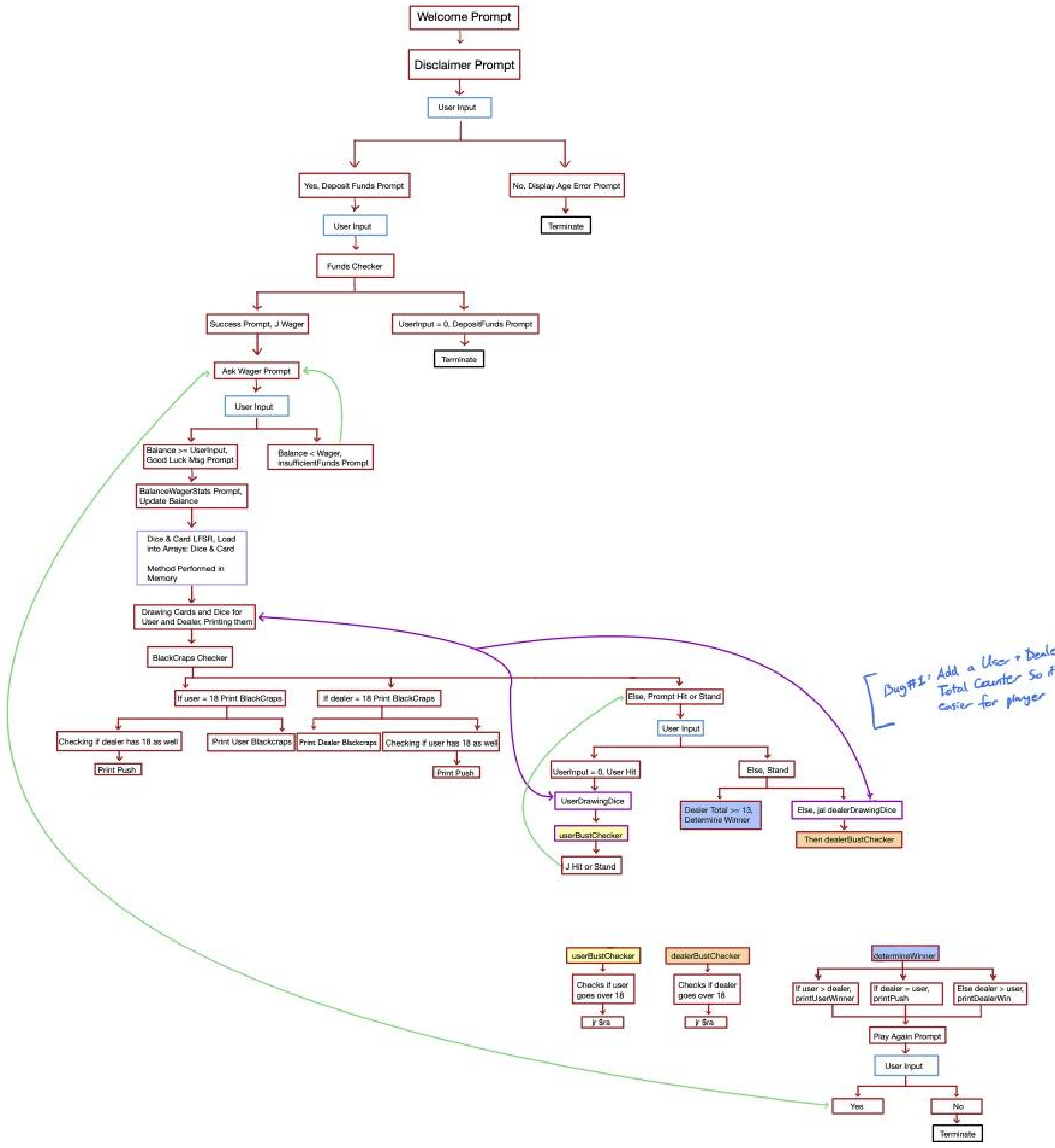
INTRODUCTION

This game derives from the game of Blackjack, but of course, with my own twist. It is called “Blackcraps”, to which a user is drawn a card and the ability to roll dice as their hands versus a dealer. Both the user and dealer’s goal is to get as close to 18, if not at 18 to secure your best odds to win against the dealer. This game involves wagering money where throughout the game, it will be tracked, updated, and stored in memory. What makes this game even more unique is the ability for the user to choose whether he wants to hit from the stack of cards or to roll the dice once more; creating a more fun, but daunting gamble in hopes to increase your chances to win against the dealer!

FLOWCHART DESIGN

Pseudocode on how the program works is included in comments within the .asm file.

BLACKCRAPS



Welcome Prompt
Disclaimer Prompt
User Input
Yes, Deposit Funds Prompt
No, Display Age Error Prompt
User Input
Terminate
Funds Checker
Success Prompt, J Wager
UserInput = 0, DepositFunds Prompt
Terminate
Ask Wager Prompt
User Input
Balance >= UserInput, Good Luck Msg Prompt
Balance < Wager, insufficientFunds Prompt
BalanceWagerStats Prompt, Update Balance
Dice & Card LFSR, Load into Arrays: Dice & Card
Method Performed in Memory
Drawing Cards and Dice for User and Dealer, Printing them
BlackCraps Checker
If user = 18 Print BlackCraps
If dealer = 18 Print BlackCraps
Else, Prompt Hit or Stand
Checking if dealer has 18 as well
Print User Blackcraps
Print Dealer Blackcraps
Checking if user has 18 as well
Print Push
Print Push
User Input
UserInput = 0, User Hit
Else, Stand
UserDrawingDice
Dealer Total >= 13, Determine Winner
Else, jal dealerDrawingDice
userBustChecker
Then dealerBustChecker
J Hit or Stand
Bug#1: Add a User + Deale
Total Counter So i
easier for player
userBustChecker
Checks if user goes over 18
jr $ra
dealerBustChecker
Checks if dealer goes over 18
jr $ra
determineWinner
If user > dealer, printUserWinner
If dealer = user, printPush
Else dealer > user, printDealerWin
Play Again Prompt
User Input
Yes
No
Terminate

Above is my final modification of my flow chart diagram implementing the Blackcraps game which helped me design bits of each code, where I was able to connect them at the end like a puzzle piece. This flow chart allows anyone how this game works and all the branches / possibilities you are expected to get within the defined code.

SOURCE CODE (ASSEMBLY)

```
.data
	lfsr:			.word			0
	cards:			.word			120
	dice:			.word			120


	pitch:			.word			72
	duration:		.word			1000
	introMusic:		.word			67, 67, 67, 66, 66, 67, 67, 67, 69, 67, 67, 67, 66, 66, 67, 67,
67, 69, 64, 64, 67, 69, 69, 67



	welcome:		.asciiz			"WELCOME TO ONLINE BLACKCRAPS!!!\nBY AJ
RODRIGO\n"
	disclaimer:		.asciiz			"\nAre you over the legal age of 21? '0' for yes, '1' for
no.\t"
	displayAgeError:	.asciiz		"\n\nUnfortunately, you have to be over the age of 21 under legal
jurisdiction. Please exit now.\n\n"
	depositFunds:		.asciiz			"\nHow much money do you wish to deposit? $"
	successfulTransfer:	.asciiz		"\nYou have successfully deposited: $"
	youAreBroke:		.asciiz		"\nPlease deposit funds.\n"
	wagerMsg:		.asciiz		"\n\nPlease place down your wager(s): $"
	zeroFunds:		.asciiz			"You have run out of funds, thank you for playing!\n\n"
	goodluck:		.asciiz		"\nThank you, GOOD LUCK!\n\n\n"
	insufficientFunds: .asciiz		"\nThe amount you wagered exceeds your balance. Please re-enter your
wager."
```

```
	hitOrStandDice:		.asciiz				"Hit or Stand? '0' for hit, '1' for stand\t"



	userCardPrint:		.asciiz				"User Card: "
	dealerCardPrint:	.asciiz			"Dealer Card: "
	userDicePrint:		.asciiz				"User Dice: "
	dealerDicePrint:	.asciiz			"Dealer Dice: "
	userTtl:		.asciiz			"User Total: "
	dealerTtl:		.asciiz				"\t\t\tDealer Total: "
	userBJ:			.asciiz				"Blackcraps! You win!\n"
	dealerBJ:		.asciiz			"Dealer has Blackcraps! You lose!\n"
	tieMsg:			.asciiz				"Push!\n"
	dealerBustMsg:		.asciiz			"Dealer has bust, You win!\n"
	userBustMsg:		.asciiz				"Bust! Dealer wins!\n"
	dealerWinMsg:		.asciiz				"Dealer wins!\n"
	userWinnerMsg:		.asciiz				"User wins!\n"
	congratsWager:		.asciiz				"Congratulations, you won $"


	diceOrCards:		.asciiz				"Would you like to draw a card or roll a dice? '0' for
cards, '1' for dice\t"
	hitStandPrompt:		.asciiz				"Hit or Stand? '0' for Hit, '1' for Stand\t"
	playAgain:		.asciiz				"Play again? '0' for Yes, '1' for No\t"


	balanceDis:		.asciiz			"\nBALANCE: $"
	wagerDis:		.asciiz			"\t\t\t\t\tWAGER: $"


	newLine:		.asciiz			"\n"
	space:			.asciiz			"\n\n"
```

```
##########################################################################################################
########################################

.text


#This section is the first part of the project to where it welcomes the user, checks if the user is of legal age, deposits funds,
checks if funds deposited is

#reasonable, asks for a wager, and displays the stats before moving onto the next section of the project


###########################

# s2 - user funds counter #

# s3 - user wager amount  #

###########################




intro:                                  #Game introduction

        addi $t4, $zero, 0

        addi $t5, $zero, 0          #Initializing temporary registers for array counter for intro song

        addi $t6, $zero, 0


        li $v0, 4

        la $a0, welcome                 #Printing welcome prompt

        syscall


        j themeSong                     #Playing sound of intro chime


disclaimerNote:

        li $v0, 4

        la $a0, disclaimer          #Printing disclaimer prompt
        syscall


        li $v0, 5
        syscall                             #Getting user input in response to disclaimer prompt
        move $t0, $v0



#If(userInput != 0)
#       printf(underTwentyOne);
#       exit();
#Else
#       printf(depositFunds);
#       scanf("%d", &x);
#
#       If(x == 0)
#               printf(youAreBroke);
#               exit();
#       Else
#               printf(success);
#               printf(" %d", x);


        addi $t1, $zero, 0
        bne $t0, $t1, underTwentyOne        #If else statement checking whether user is over the legal age
        j depositingFunds


underTwentyOne:                             #If user is not over 21 then come to this routine, which will give user
feedback and terminate
        li $v0, 4
        la $a0, displayAgeError             #Printing displayAgeError message
        syscall

        j end                               #Terminates program

depositingFunds:                    #If user is over the age of 21, then it will jump to this routine and skip the
underTwentyOne routine
        li $v0, 4
        la $a0, depositFunds                #Printing depositFunds message
        syscall

        li $v0, 5
        syscall                             #Getting user input for game funds
        move $s2, $v0

        j fundsChecker

fundsChecker:                               #Checking funds if amount of funds entered is greater than zero
        bnez $s2, success
        li $v0, 4
        la $a0, youAreBroke                 #Printing youAreBroke message
        syscall
        j end

success:                            #If user input was greater than zero, then come to this routine
        li $v0, 4
        la $a0, successfulTransfer  #Printing successfulTranser message
        syscall

        jal positiveSound           #Function for sound feedback successful deposit chime
        move $a0, $s2
        li $v0, 1                   #Printing the value of the user input as confirmation
        syscall
        j wager

#printf(wagerMsg);
#scanf("%d", &y);
#While(x < y)
#       printf(insufficientFunds);
#printf(goodLuck);
#printf(balanceDis);
#x -= y;
#printf(" %d", &x);
#printf(wagerDis);
#printf(" %d", &y);

wager:
        li $v0, 4
        la $a0, wagerMsg            #Printing wagerMsg prompt
        syscall

        li $v0, 5
        syscall                             #Getting user input for amount of wager to play with and move it to the
saved register
        move $s3, $v0

        bge $s2, $s3, goodluckMessage       #Condition to check if balance is greater or equal to wager
        li $v0, 4
        la $a0, insufficientFunds    #Printing insufficientFunds message
        syscall

        jal negativeSound            #Function for sound feedback invalid wager chime

        j wager                              #Loop until wager is within the conditions

goodluckMessage:
        li $v0, 4
        la $a0, goodluck             #Printing goodLuck message as successful feedback before printing out the
balanceWagerStats
        syscall

balanceWagerStats:
        li $v0, 4
        la $a0, balanceDis           #Printing balanceDis message
        syscall

        sub $s2, $s2, $s3            #Updating balance after accepting user inputted wager

        move $a0, $s2
        li $v0, 1                    #Printing new total balance after update
        syscall

        li $v0, 4
        la $a0, wagerDis             #Printing wagerDis message
        syscall
```

```
	move $a0, $s3

	li $v0, 1			#This will print the amount wagered

	syscall


	li $v0, 4

	la $a0, space				#Printing space for tidiness

	syscall


	addi $t8, $zero, 0		#Initiallizing the t8 register which will serve as the register that holds the cards
pulled


################################################################################################
#####################################


# 32 Bit LFSR for both cards and dice

# Partial code used from Homework 3


cardMain:

	addi $t9, $zero, 0		#Initializing t9 register that will serve as a counter to store the dice and card array
with random numbers

	la $t7, cards				#Loading address of cards to t7 register

	jal cardsFirst


cardsFirst:

	addi $sp, $sp, -4		#Multiple function calling, saving address of this function to stack

	sw $ra, 0($sp)

	addi $t1, $zero, 0

	beq $t9, 30, diceMain			#Counter condition that will populate card array with 30 numbers before
populating dice array

	j lfsrNum			#LFSR function that will get random numbers
```

```
diceMain:
        addi $t5, $zero, 0x10010084  #Loading t5 with address to store the next 30 random numbers into dice array
        jal diceSecond

diceSecond:
        addi $sp, $sp, -4            #Saving diceSecond address so that it can loop and populate dice array
        sw $ra, 0($sp)
        addi $t1, $zero, 0
        beq $t9, 60, drawingMain     #Counter condition that will populate dice array with the next 30 numbers
        j lfsrDice

lfsrDice:                            #LFSR dice routine
        la $t0, lfsr
        jal randNumGen
        beq $t1, 16, diceRoll
        j randNumGen

lfsrNum:                             #LFSR card routine
        la $t0, lfsr
        jal randNumGen
        beq $t1, 16, deckOfCards


#If(a0 == 0)
#       int a0 = 0xF00F5AA5
#       lfsr[] = a0;
#int t2 = 1;
```

```
#int s0 = a0 & t2;

#int t3 = a0 << 2;

#int s1 = t3 & t2;

#int s4 = s0 ^ s1;

#int t3 = 0;

#t3 = a0 << 7;

#s1 = t3 & t2

#int s5 = s0 ^ s1;

#int s6 = s4 ^ s5;

#

#If(s6 == 0)

#           a0 << 1;

#           lfsr[] = a0;

#           t1 += 1;

#Else

#           int t4 = 0x80000000;

#           a0 << 1

#           a0 = a0 + t4;

#           lfsr[] = a0;

#           t1 += 1;


#Used from homework 3, slightly modified

randNumGen:                                         #Random number generator LFSR

            lw $a0, 0($t0)

            beqz $a0, initialSeedValue    #If LFSR array is not initialized, then go ahead and initialize with seed value


            addi $t2, $zero, 1             #Mask number one that will strip the 32nd bit

            and $s0, $a0, $t2              #Bitwise and that will get the 32nd bit and store it in s0
```

```
	srl $t3, $a0, 2				#Shifting LFSR array by 2 to get the 30th bit

	and $s1, $t3, $t2


	xor $s4, $s0, $s1			#Bitwise XOR of the 32nd and 30th bit we had just stripped


	addi $t3, $zero, 0		#Resetting register

	srl $t3, $a0, 6				#Shifting to get the 26th bit of seed

	and $s0, $t3, $t2


	addi $t3, $zero, 0		#Resetting register

	srl $t3, $a0, 7				#Shifting to get the 25th bit of seed

	and $s1, $t3, $t2


	xor $s5, $s0, $s1			#Bitwise XOR of 26th and 25th bit we had just stripped


	xor $s6, $s4, $s5			#Bitwise XOR of the results of XOR of 32nd and 30th bit and XOR of 25th and 26th bit


	beqz $s6, addingZero			#Checking if the XOR results is equal to 0, if it is then shift number once to
add a zero to front

	addi $t4, $zero, 0x80000000 #Creating a mask that will add 1 to the front of the original 32 bit number

	srl $a0, $a0, 1				#Shifting by one and adding that to original number to get a one in front

	add $a0, $a0, $t4

	sw $a0, 0($t0)

	addi $t1, $t1, 1			#Updating count


	jr $ra					#Jumps back to either lfsrNum or lfsrDice routine depending on what
function called it
```

```
addingZero:                             #This routine shifts 32 bit number, placing 0 in front and storing to array,
and updating count

        srl $a0, $a0, 1

        sw $a0, 0($t0)

        addi $t1, $t1, 1



        jr $ra



initialSeedValue:                       #Seed initializing value to this 32 bit number

        addi $a0, $zero, 0xF00F5AA5

        sw $a0, 0($t0)

        j lfsrNum



#DECK OF CARDS#

#int t6 = 0xF;

#t8 = lfsr[t0] & t6;

#If(t8 > 11)

#       t8 -= 4;

#       cards[t7] = t8;

#       t7 += 1;

#       t9 += 1;

#       cardsFirst();

#Else if(t8 == 0)

#       t8 += 4;

#       cards[t7] = t8;

#       t7 += 1;

#       t9 += 1;

#       cardsFirst();

#Else
```

```
#		cards[t7] = t8;

#		t7 += 1;

#		t9 += 1;

#		cardsFirst();



deckOfCards:

		addi $t6, $zero, 0xF				#Mask that will limit numbers from 1-11 for cards

		lw $a0, 0($t0)

		and $t8, $a0, $t6			#Strips 32 bit number down to the last 4 bits

		bgt $t8, 11, subtract4				#Condition that will determine if we need to trim the number if number is
over 11

		beqz $t8, add4					#If the number is 0, then we will make the number 4

		sw $t8, 0($t7)

		addi $t7, $t7, 4			#Updating array location, 4 bytes

		addi $t9, $t9, 1			#Counter for populating array


		lw $ra, 0($sp)					#Popping array using stack to return to function call

		addi $sp, $sp, 4

		jr $ra


#DECK OF DICE#

#int t6 = 0x7;

#t8 = lfsr[t0] + t6;

#If(t8 == 0)

#		t8 += 3;

#		dice[t5] = t8;

#		t5 += 1;

#		t9 += 1;
```

```
#		diceSecond();

#Else

#		dice[t5] = t8;

#		t5 += 1;

#		t9 += 1;

#		diceSecond();



diceRoll:

		addi $t6, $zero, 0x7				#Mask that will limit numbers from 1-11 for cards

		lw $a0, 0($t0)

		and $t8, $a0, $t6			#Strips 32 bit number to the last 3 bits

		beqz $t8, add3				#Conditions that checks if new number is zero, then that number will
become 3

		sw $t8, 0($t5)				#Store new number to dice array

		addi $t5, $t5, 4			#Updating array location, 4 bytes

		addi $t9, $t9, 1			#Counter for populating array


		lw $ra, 0($sp)				#Popping array using stack to return to function call

		addi $sp, $sp, 4

		jr $ra


subtract4:					#This function subtracts 4 if number is greater than 11 for cards puller,
stores into cards array, updates counter					#

		sub $t8, $t8, 4				#and jumps backs to function using popping with stack

		sw $t8, 0($t7)

		addi $t7, $t7, 4

		addi $t9, $t9, 1
```

```
        lw $ra, 0($sp)

        addi $sp, $sp, 4

        jr $ra


add3:                                   #This function adds 3 to dice array if new number stripped is zero, stores
into dice array, updates counter

        add $t8, $t8, 3                 #and jumps backs to function using popping with stack

        sw $t8, 0($t5)

        addi $t5, $t5, 4

        addi $t9, $t9, 1


        lw $ra, 0($sp)

        addi $sp, $sp, 4

        jr $ra


add4:                                   #This function adds 4 to cards array if new number is zero, stores into
cards array, updates counter

        add $t8, $t8, 4                 #and jumps backs to function using popping with stack

        sw $t8, 0($t7)

        addi $t7, $t7, 4

        addi $t9, $t9, 1


        lw $ra, 0($sp)

        addi $sp, $sp, 4

        jr $ra

##########################################################################################################
########################################


###########################################
```

```
# t0 - start of card array                #
# t1 - start of dice array                #
# t6 - dealer number running total        #
# t8 - user number pulled running total   #
#############################################

drawingMain:                                    #Initializing some temporary registers
        addi $t0, $zero, 0
        addi $t1, $zero, 124
        addi $t3, $zero, 0
        addi $t6, $zero, 0
        addi $t7, $zero, 0
        addi $t8, $zero, 0
        addi $t9, $zero, 0

drawingGameFlowCards:                           #Drawing both user and dealer cards
        jal userDrawingCards
        jal dealerDrawingCards
        j drawingGameFlowDice

drawingGameFlowDice:                            #Drawing both user and dealer dice rolls
        beq $t3, 1, blackcrapsChecker           #Once both hands drawn, check if anyone got blackcraps
        addi $t3, $t3, 1                #Counter

        jal userDrawingDice
        jal dealerDrawingDice
        j drawingGameFlowDice
```

```
userDrawingCards:
        li $v0, 4
        la $a0, userCardPrint               #Printing userCardPrint message
        syscall

        lw $t9, cards($t0)
        move $a0, $t9                       #Loading user number to register and displaying it
        li $v0, 1
        syscall

        addi $t0, $t0, 4            #Updating cards array location, 4 bytes

        li $v0, 4
        la $a0, newLine                     #Printing newLine for tidiness
        syscall

        add $t8, $t8, $t9           #Loading pull into t8 which will actively keep track of user running total
        jr $ra

dealerDrawingCards:
        li $v0, 4
        la $a0, dealerCardPrint             #Printing dealerCardPrint message
        syscall

        lw $t9, cards($t0)
        move $a0, $t9                       #Loading dealer number to register and displaying it
        li $v0, 1
        syscall
        addi $t0, $t0, 4              #Updating cards array location, 4 bytes


        li $v0, 4

        la $a0, newLine                      #Printing newLine for tidiness

        syscall


        add $t6, $t6, $t9             #Loading pull into t6 which will actively keep track of dealer running total

        jr $ra


userDrawingDice:

        li $v0, 4

        la $a0, userDicePrint                #Printing userDicePrint message

        syscall


        lw $t7, dice($t1)

        move $a0, $t7                        #Loading user dice number to register and displaying it

        li $v0, 1

        syscall


        addi $t1, $t1, 4              #Updating cards array location, 4 bytes


        li $v0, 4

        la $a0, newLine                      #Printing newLine for tidiness

        syscall


        add $t8, $t8, $t7             #Loading dice pull into t8 which will actively keep track of user running total

        jr $ra
```

```
dealerDrawingDice:

        li $v0, 4

        la $a0, dealerDicePrint                 #Printing dealerDicePrint

        syscall


        lw $t7, dice($t1)

        move $a0, $t7                           #Loading dealer dice number to register and displaying it

        li $v0, 1

        syscall


        addi $t1, $t1, 4                #Updating cards array location, 4 bytes


        li $v0, 4

        la $a0, newLine                         #Printing newLine for tidiness

        syscall


        add $t6, $t6, $t7               #Loading dice pull into t6 which will actively keep track of dealer running total

        jr $ra

##################################################################################################
########################################


#This section contains code for how game functions / game rules


#If(user == 18)

#       if(dealer == 18)

#               printf(printPush)

#       printf(userBJ)
```

```
#		t7 = s2 * 2;

#		s2 += t7;

#		printf(balanceDis);

#		printf("%d", s2);

#		printf(space);

#		goto playAgainPrompt;

#Else if(dealer == 18)

#		if(user == 18)

#				printf(printPush);

#		printf(dealerBJ)

#		printf(balanceDis);

#		printf("%d", s2);

#		printf(space);

#		goto playAgainPrompt;

#Else

#		goto hitOrStand;


blackcrapsChecker:					#Checks if either user or dealer has Blackcraps on first hand

		beq $t8, 18, userBlackcrapsPrint

		beq $t6, 18, dealerBlackcrapsPrint

		j hitOrStand


userBlackcrapsPrint:

		beq $t6, 18, printPush			#If user has 18 and dealer has 18 on first hand, then print push


		li $v0, 4

		la $a0, userBJ				#Printing userBJ message

		syscall
```

```
	mul  $t7, $s3, 2		#Paying out user by multiplying wager by two and adding it to the remaining
balance
	add $s2, $s2, $t7

	li $v0, 4
	la $a0, balanceDis		#Printing balanceDis message
	syscall

	move $a0, $s2
	li $v0, 1			#Printing new balance after user wins
	syscall

	li $v0, 4
	la $a0, space				#Printing space for tidiness
	syscall

	j playAgainPrompt		#Loop that prompts user if he wants to play again

dealerBlackcrapsPrint:
	beq $t8, 18, printPush			#If dealer has 18 and user has 18 on first hand, then print push

	li $v0, 4
	la $a0, dealerBJ		#Printing dealerBJ message
	syscall

	li $v0, 4
	la $a0, balanceDis		#Printing balanceDis message
	syscall
	move $a0, $s2
	li $v0, 1			#Printing new balance after dealer wins
	syscall

	li $v0, 4
	la $a0, space				#Printing space for tidiness
	syscall

	j playAgainPrompt		#Loop that prompts user if he wants to play again

#printPush()
#	printf(tieMsg);
#	printf(balanceDis);
#	s2 += s3;
#	printf("%d", s2);
#	printf(space);
#	goto playAgainPrompt;
#	return;

printPush:
	li $v0, 4
	la $a0, tieMsg				#Printing tieMsg
	syscall

	li $v0, 4
	la $a0, balanceDis		#Printing balanceDis message
	syscall
	add $s2, $s2, $s3
	move $a0, $s2				#Tie = no money loss, adding wager back to balance and displaying it
	li $v0, 1
	syscall

	li $v0, 4
	la $a0, space				#Printing space for tidiness
	syscall

	j playAgainPrompt		#Loop that prompts user if he wants to play again

#hitOrStand();
#	printf(space);
#	printf(userTtl);
#	printf("%d", t8);
#	printf(dealerTtl);
#	printf("%d", t6);
#	printf(space);
#	printf(hitStandPrompt);
#	scanf("%d", &t7);
#	If(t7 == 0)
#		goto userHit;
#	Else
#		goto stand;

hitOrStand:
	li $v0, 4
```

```
la $a0, space                    #Printing space for tidiness
syscall

li $v0, 4
la $a0, userTtl                  #Printing out userTtl design
syscall

move $a0, $t8
li $v0, 1               #Printing user running total number
syscall

li $v0, 4
la $a0, dealerTtl       #Printing dealerTtl design
syscall

move $a0, $t6
li $v0, 1               #Printing dealer running total number
syscall

li $v0, 4
la $a0, space                    #Printing space for tidiness
syscall

li $v0, 4
la $a0, hitStandPrompt           #Printing hitStandPrompt
syscall

li $v0, 5
```

```
        syscall                         #Getting user input whether user wants to hit or stand
        move $t7, $v0

        beqz $t7, userHit           #If user inputs zero, then hit, else stand
        j stand

#userHit()
#       printf(diceOrCards);
#       scanf("%d", &t7);
#       printf(newLine);
#       If(t7 == 0)
#               userDrawingDice();
#               userBustChecker();
#               goto hitOrStand;
#       Else
#               userDrawingDice();
#               userBustChecker();
#               goto hitOrStand;

userHit:
        li $v0, 4
        la $a0, diceOrCards             #Printing diceOrCards prompt
        syscall

        li $v0, 5
        syscall                         #Getting user input in response to whether user would like cards to draw
or dice to roll; it's your gamble
        move $t7, $v0
        li $v0, 4
        la $a0, newLine                 #Printing newLine for tidiness
        syscall

        beqz $t7, userCardsOpt          #If user chooses to hit cards then go to this routine
        jal userDrawingDice             #Else user draws dice and always check after drawing whether user busts
        jal userBustChecker
        j hitOrStand                    #Loop hit or stand prompt

userCardsOpt:                           #Function that draws user cards upon request of drawing, always checks if
user busts, loop hit or stand prompt
        jal userDrawingCards
        jal userBustChecker
        j hitOrStand

#If(t6 >= 13)
#       goto determineWinner;
#Else
#       printf(newLine);
#       dealerDrawingDice();
#       goto dealerBustChecker;

stand:
        bge $t6, 13, determineWinner    #When you stand, check if dealer has at least a 13 before determining the
winner. Dealers stands on a 13

        li $v0, 4
        la $a0, newLine                 #Printing newLine for tidiness
        syscall
        jal dealerDrawingDice               #Dealer automatically draws dice as his hit cards and followed by a dealer
bust checker

        j dealerBustChecker


#userBustChecker()

#       int t5 = 18;

#       If(user > 18)

#               printf(userBustMsg);

#               printf(balanceDis);

#               printf("%d", s2);

#               printf(space);

#               goto playAgainPrompt;

#       return;


userBustChecker:                    #Checks if user goes past 18

        addi $t5, $zero, 18

        blt $t5, $t8, displayUserBust

        jr $ra


#dealerBustChecker()

#       int t5 = 18;

#       If(dealer > 18)

#               printf(dealerBustMsg);

#               t7 = s3 * 2;

#               s2 += t7;

#               printf(balanceDis);

#               printf("%d", s2);

#               printf(space);
#                       goto playAgainPrompt;

#           return;



dealerBustChecker:                          #Checks if dealer goes past 18

            addi $t5, $zero, 18

            blt $t5, $t6, displayDealerBust

            j stand


#determineWinner()

#           printf(newLine);

#           printf(space);

#           printf(userTtl);

#           printf("%d", t8);

#           printf(dealerTtl);

#           printf("%d", t6);

#           printf(space);

#           If(user > dealer)

#                       printf(userWinnerMsg);

#                       printf(congratsWager);

#                       printf("%d", s3);

#                       printf(space);

#                       t7 = s3 * 2;

#                       s2 += t7;

#                       printf(balanceDis);

#                       printf("%d", s2);

#                       printf(space);

#                       goto playAgainPrompt
```

```
#		Else if(dealer == user)
#				goto printPush;
#		Else
#				printf(dealerWinMsg);
#				printf(balanceDis);
#				printf("%d", s2);
#				printf(space);
#				goto playAgainPrompt;

determineWinner:				#Determining winner by comparing the dealer and user total, if they are equal, then
print push
		li $v0, 4
		la $a0, newLine				#Printing newLine for tidiness
		syscall

		li $v0, 4
		la $a0, space				#Printing space for tidiness
		syscall

		li $v0, 4
		la $a0, userTtl				#Printing out userTtl design
		syscall

		move $a0, $t8
		li $v0, 1			#Printing user running total number
		syscall

		li $v0, 4
		la $a0, dealerTtl		#Printing dealerTtl design
```

```
syscall


move $a0, $t6

li $v0, 1                              #Printing dealer running total number

syscall


li $v0, 4

la $a0, space                                  #Printing space for tidiness

syscall


bgt $t8, $t6, printUserWinner

beq $t8, $t6, printPush


li $v0, 4

la $a0, dealerWinMsg                   #Printing dealerWinMsg if dealer > user

syscall


li $v0, 4

la $a0, balanceDis             #Printing balanceDis message design

syscall


move $a0, $s2

li $v0, 1                              #Printing out new balance after losing your hand

syscall


li $v0, 4

la $a0, space                                  #Printing space for tidiness

syscall
```

```
        j playAgainPrompt           #Loop play again prompt


printUserWinner:
        li $v0, 4
        la $a0, userWinnerMsg               #Printing userWinnerMsg if user > dealer
        syscall

        li $v0, 4
        la $a0, congratsWager               #Printing congratsWager message
        syscall

        move $a0, $s3
        li $v0, 1                   #Printing amount you wagered
        syscall

        li $v0, 4
        la $a0, space                       #Printing out space for tidiness
        syscall

        mul  $t7, $s3, 2            #Paying out user by multiplying wager by two and adding it to the remaining
balance
        add $s2, $s2, $t7

        li $v0, 4
        la $a0, balanceDis          #Printing out balanceDis message design
        syscall

        move $a0, $s2
        li $v0, 1                   #Printing out new updated balance after user win
        syscall

        li $v0, 4
        la $a0, space                       #Printing out space for tidiness
        syscall

        j playAgainPrompt           #Loop play again prompt

displayDealerBust:
        li $v0, 4
        la $a0, dealerBustMsg               #Printing dealerBustMsg
        syscall

        mul  $t7, $s3, 2            #Paying out user by multiplying wager by two and adding it to the remaining
balance
        add $s2, $s2, $t7

        li $v0, 4
        la $a0, balanceDis          #Printing out balanceDis message design
        syscall

        move $a0, $s2
        li $v0, 1                   #Printing out new updated balance after user win
        syscall

        li $v0, 4
        la $a0, space                       #Printing out space for tidiness
        syscall
```

```
	j playAgainPrompt			#Loop play again prompt

displayUserBust:
	li $v0, 4
	la $a0, userBustMsg			#Printing userBustMsg
	syscall

	li $v0, 4
	la $a0, balanceDis		#Printing out balanceDis message design
	syscall

	move $a0, $s2
	li $v0, 1				#Printing out new user total after losing hand
	syscall

	li $v0, 4
	la $a0, space				#Printing out space for tidiness
	syscall

	j playAgainPrompt			#Loop play again prompt

#playAgainPrompt()
#	If(s2 == 0)
#		printf(newLine);
#		printf(zeroFunds);
#		exit();
#	printf(playAgain);
```

```
#		scanf("%d", t7);
#		If(t7 == 0)
#				goto wager;
#		Else
#				exit();

playAgainPrompt:
		beqz $s2, youHitZero			#Checking whether your total balance is still above zero
		li $v0, 4
		la $a0, playAgain		#Printing playAgain message
		syscall

		li $v0, 5
		syscall					#Getting user input whether if user wants to play another game
		move $t7, $v0

		beqz $t7, wager				#If user wants to play again, jump to wager to prompt process all over
again
		j end					#Else terminate

youHitZero:
		li $v0, 4
		la $a0, newLine				#Printing newLine for tidiness
		syscall

		li $v0, 4
		la $a0, zeroFunds		#Printing out zeroFunds message
		syscall
```

```
	j end					#When user has no more funds left, terminate program


###################################################################################################
#####################################

#SOUND FX

positiveSound:

	lw $t1, pitch($zero)			#Loading integer from first element in array to temp registers

	lw $t2, duration($zero)


	li $v0, 31			#31 Syscall for sound

	la $a0, ($t1)				#Loading into a0 for pitch

	la $a1, ($t2)				#Loading into a1 for duration

	la $a2, 33			#Loading number to a2 for instrument

	la $a3, 127				#Loading number to a3 for volume

	syscall


	jr $ra


negativeSound:				#Same as above, but with a different instrument

	lw $t1, pitch($zero)

	lw $t2, duration($zero)


	li $v0, 31

	la $a0, ($t1)

	la $a1, ($t2)

	la $a2, 113

	la $a3, 127

	syscall
	jr $ra


themeSong:				#Function plays notes in array as theme song, once it plays the array then it
will continue with prompt of the game

	beq $t4, 24, disclaimerNote


	lw $t1, introMusic($t5)

	lw $t2, duration($zero)


	li $v0, 31

	la $a0, ($t1)

	la $a1, ($t2)

	la $a2, 0

	la $a3, 127

	syscall


	addi $t4, $t4, 1		#Counter for each note

	addi $t5, $t5, 4		#Counter for moving element in array, 4 bytes


	li $v0, 32

	addi $a0, $zero, 220

	syscall


	j themeSong


##########################################################################################################
########################################

end:

	jal negativeSound
```

```
li $v0, 10                    #Terminating program

syscall
```

TOOLS USED

1. Mars 4.5
    a. Syscall functions
    b. Arrays
    c. Stack address
    d. MIDI out sound

USER INSTRUCTIONS

1. Upon run, follow the introduction prompts to make sure that user is of legal age
2. Once successful, please deposit money
    a. If user enters $0, then it will prompt you to enter a correct amount
3. Amount deposited will allow you to place down wager
    a. If deposit is accepted, it will save your deposit throughout the game
    b. Wager placed has to be equal or less than you deposit
        i. If you deposit $50 and wager $50, that is valid
        ii. If you deposit $50 and wager $20, that is valid
        iii. If you deposit $50 and wager $60, that is invalid and will prompt you to enter a valid wager
4. After your wager has been accepted by the casino, the computer will then proceed to draw and roll dice for both the dealer and user playing
    a. You will see the total number of dealer and user, in order for the user to make his next decision
5. There will be a prompt whether to Hit or Stand and this is solely the users job to come up with his decision assuming that he knows the rules and the number before busting
    a. If user decides to choose to hit, user will given the option to choose whether his next draw would be from the deck of cards or through the dice

roll

        i. Clearly up to user if he wants to risk dice roll or draw cards
            1. Cards have values from 1-11
                a. It would be smart to hit from hards if you have a hand lower than 7
                    i. If user has a 7, you would want to hit from cards due to the fact that there is a possibility that you can get yourself closer to the 18 by landing a 11 or a 10, which can only be pulled from the deck of cards
            2. Dice rolls have values from 1-7
                a. It would be smart to hit from dice if you have anything above a 7
                    i. If user has 15, you would not want to hit from cards user has a higher chance of busting since there are more high numbers

6. Upon stand, everything will be calculated in real time to which the computer will determine the winner
    a. If there is no user nor dealer Blackcraps, then it will prompt to hit or stand
    b. Everytime the user hits, we will check whether he has busted or not
    c. When user stands, dealer keeps hitting until he reaches soft 13 to which the computer would then determine the winner based on who has the greater number
        i. If user has a 15 and dealer has 13, user wins
        ii. If user has 13 and dealer has 15, dealer wins
        iii. If at any given moment user goes over 18, dealer wins
        iv. If at any given moment dealer goes over 18, user wins
        v. If dealer and user both have the same number after standing, then it is a push, no one wins
7. After determining a winner and updating the user's new balance, user will be prompted to play again
    a. If user has no more funds, then end game
        i. If user has $0 at the end of the game then kick him out of casino
        ii. If user has $x amount of dollars left, prompt user if he wants to play again
        iii. If user does have $x amount of dollars left, but does not want to play again, then terminate game

RESULTS - TEST RUNS

1. A successful run

```
WELCOME TO ONLINE BLACKCRAPS!!!

BY AJ RODRIGO

Are you over the legal age of 21? '0' for yes, '1' for no.        0

How much money do you wish to deposit? $50

You have successfully deposited: $50

Please place down your wager(s): $50

Thank you, GOOD LUCK!


BALANCE: $0                                          WAGER: $50


User Card: 10

Dealer Card: 5

User Dice: 2

Dealer Dice: 5


User Total: 12                    Dealer Total: 10


Hit or Stand? '0' for Hit, '1' for Stand        0

Would you like to draw a card or roll a dice? '0' for cards, '1' for dice        1


User Dice: 5

User Total: 17                    Dealer Total: 10


Hit or Stand? '0' for Hit, '1' for Stand        1


Dealer Dice: 3

User Total: 17                    Dealer Total: 13
User wins!

Congratulations, you won $50


BALANCE: $100


Play again? '0' for Yes, '1' for No    0


Please place down your wager(s): $100


Thank you, GOOD LUCK!


BALANCE: $0                                        WAGER: $100


User Card: 7

Dealer Card: 6

User Dice: 7

Dealer Dice: 6



User Total: 14                         Dealer Total: 12


Hit or Stand? '0' for Hit, '1' for Stand          0

Would you like to draw a card or roll a dice? '0' for cards, '1' for dice          1


User Dice: 7

Bust! Dealer wins!


BALANCE: $0

You have run out of funds, thank you for playing!
```

```
-- program is finished running --
```

2. Unsuccessful run due to age

```
WELCOME TO ONLINE BLACKCRAPS!!!

BY AJ RODRIGO

Are you over the legal age of 21? '0' for yes, '1' for no.      1

Unfortunately, you have to be over the age of 21 under legal jurisdiction. Please exit now.

-- program is finished running --
```

3. Unsuccessful run from depositing $0

```
WELCOME TO ONLINE BLACKCRAPS!!!

BY AJ RODRIGO

Are you over the legal age of 21? '0' for yes, '1' for no.      0

How much money do you wish to deposit? $0

Please deposit funds.

-- program is finished running --
```

DOWNLOAD LINK OF ASM FILE

ASM file:

https://drive.google.com/file/d/1M708QbQ6MyR8N6zeMNVFdIOoD3zlzk1f/view?usp=sharing

.DOC file of source code:

https://drive.google.com/file/d/1cB-k7Ay4K1n0tXT_BKOk765_waYxKFp8/view?usp=sharing

PROJECT LIMITATIONS, BUGS, NOTES

As far as bugs, there is one that may catch one's eye if they accidentally input a value other than 0 or 1 as the user's yes or no. Any value other than zero, will be accounted for no because of the way the program is coded. If I wanted to fix this bug, I would have to add another if condition to test if input == 1, then branch to no. While else, if neither 0 or 1 is entered by the user, have it repeat and loop the question until input is valid.

There may be other bugs that I have yet to catch, but that is the only one so far that stands out.

I had trouble implementing the intro to continue to play a theme chime as the “lobby music” until user was able to verify if he was of legal age, but I couldn’t get the sound to play in the background as it had to finish playing the notes in the array before moving on to the next function.

Project notes before getting approved (as original idea was Blackjack):
https://drive.google.com/file/d/1ycvUi5s_NSiL1qkrpwZxtpB07B1lVOY9

Project notes for actual Blackcraps, not Blackjack:
https://drive.google.com/file/d/1digEBNNeRyUOilJ-Y8Ac-ReS_yCXtgIT/view?usp=drivesdk

YOUTUBE SCREEN CAPTURE LINK

YouTube Link: https://youtu.be/jjygSHbnzeY

** Please note that I used sound effects MIDI out for my project, but under the screen capture it was not able to pick up the sound effects or any audio at all. To allow the sound effects to play, please run the code on your own computer to hear the sound implemented to the program!

FINAL THOUGHTS - WHAT I WOULD DO DIFFERENTLY

What I would do differently next time is that I would try to implement graphics to this program as it would add more life and visualization. Oftentimes, it gets kind of tiring just trying to read basic Arial font text with no graphical design, which in the real world, would definitely drive a player to sleep! As far as my code that I have right now, I feel pretty accomplished and happy with the results and my progression. Able to implement all these features like the LFSR, using arrays, or even just low level coding itself, made this class fun and interesting.

CONCLUSION

After accomplishing this semester-long project that I have been practically working on since the beginning of class, I have concluded that I have learned a lot of things throughout the semester. I am more than proud of myself for being able to get through this project, a vision that I had since the beginning of project proposals, and being able to finish with a physical working program that I designed. Despite trying to take the easy route, for example originally using a 3-bit and 4-bit LFSR instead of the 32-bit LFSR just to get the project done, I was really just trying to pass by, but through long thinking and configuration, I am glad that I put in the hours in choosing to incorporate the 32-bit LFSR with my program. Everything seems hard at first when you don't have a blueprint, a game plan, but after developing steps and procedures on what I should do, I was able to overcome this tedious obstacle which I am grateful for.

HOURS SPENT

I began this project a week before class actually started as you had scared me with the email you had sent prior to the beginning of class.

I worked on bits of code at a time, never really coding everything all at once. Also I never wrote each function on the same file, as I would test and debug each function before I was able to combine the whole project at the end.

Nevertheless, I probably spent on average between four to six hours every week within the first couple months, took a break within the middle of it where I would not look at the code at all, then came back and did the same routine (4 hours per week) upon completion.

I would say I definitely spent just about or more than 45 hours on it.

REFERENCES

1. ZyBooks Chapter 3-5 - Main Learning Source
2. LFSR Section copied and modified from Compe 271 HW#3
3. Hex to Binary Converter - to help bit manipulate the LFSR
    a. https://www.rapidtables.com/convert/number/hex-to-binary.html

4. MIDI Out Section (MIPs Sound) - referenced from link that came from Compe 271 Discord Chat
    a. https://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html

5. Function call $ra - referenced from March 16, 2021 Compe 271 Lecture
    a. https://sdsu.zoom.us/rec/play/RBRf8X8nj0R5tTg8W9AsdXgfOMy3ZKLlB2cf0wosVXs1N4bBbSUDShBVVwYmbVqxNhXIhGD_19GZpe2r.iNQW2ICVLBMDhVPW?continueMode=true&_x_zm_rtaid=HtX9zqc5Q-ekV9Tkovqd5g.1618279556262.d21bfa590c94886920c8ba39f35a0c52&_x_zm_rhtaid=914