Lab 5: A Gambling Game

Due Tuesday, 4 June 2019 11:59 PM

A Plea

Read this lab assignment in its entirety before posting any questions to Piazza. For real. Actually read the friggin' manual. Please?

Thank you!
your friendly CE12 staff

Minimum Submission Requirements

- Your Lab5 folder must contain the following files (note the capitalization convention):
    - Lab5.asm (**must assemble** both **with and without** test file)
    - README.txt
- Commit and Push your repository

Lab Objective

In this lab, you will learn how to:

1. Implement subroutines
2. Manage data on the stack
3. Build an interactive game

Lab Preparation

Read

Introduction To MIPS Assembly Language Programming: chapters 5, 6; sections 8.1, 8.2

Macros

Watch

MIPS: Procedures and jal (David B): videos 7 - 12

MIPS Tutorials: Functions (Amell Peralta): videos 15 - 18

Description

In this game, an array of unique, positive, non-zero 32 bit integers will be given. The player must make a bet and guess the index of the largest element in the array. If the player guesses correctly, that element is replaced with a value of -1 and the score will increase by the amount of the bet. If the player guesses incorrectly, the score will decrease by the amount of the bet.

The player can continue to make bets and guess until they are either out of points, or all elements of the array have been guessed correctly.

Specification

You will need to implement a set of specific subroutines indicated in this lab instruction. **You are required to start with the skeleton code provided** on Canvas (lab5_template_19sp_cmpe012.asm). **Don't forget to rename** the file to Lab5.asm

A test file (lab5_test_19sp_cmpe012.asm) is provided that imports and tests each one of your subroutines individually, and all together. In order for your subroutines to function properly, **you must use** the instructions **JAL and JR** to enter and exit subroutines.

To receive **any credit** for your subroutines, **Lab5.asm must assemble** both **on its own** and with the test file.

Sample Playthroughs

The console of several playthroughs are included in the Appendix.

Skeleton Code

Start writing your subroutines using the skeleton code (lab5_template_19sp_cmpe012.asm) provided on Canvas. Make sure to **rename this file to Lab5.asm**.

Test File

The file lab5_test_19sp_cmpe012.asm contains code that you can use to test your subroutines. To test your subroutines:

1. Ensure lab5_test_19sp_cmpe012.asm is in the same folder as Lab5.asm
2. **Change the last line of the test file** from:
   ```
   .include  "lab5_template_19sp_cmpe012.asm"
   ```
   to
   ```
   .include  "Lab5.asm"
   ```
3. Assemble and run the test file.

Notes

The static data segment contains the following labels:

*score_* points to the initial score

*array_* points to the address of index 0

You can assume each element of the array is a 32-bit two's complement integer. The array is terminated with a value of 0x00000000.

Block Diagrams

Program Flow

This diagram illustrates how the program will flow.

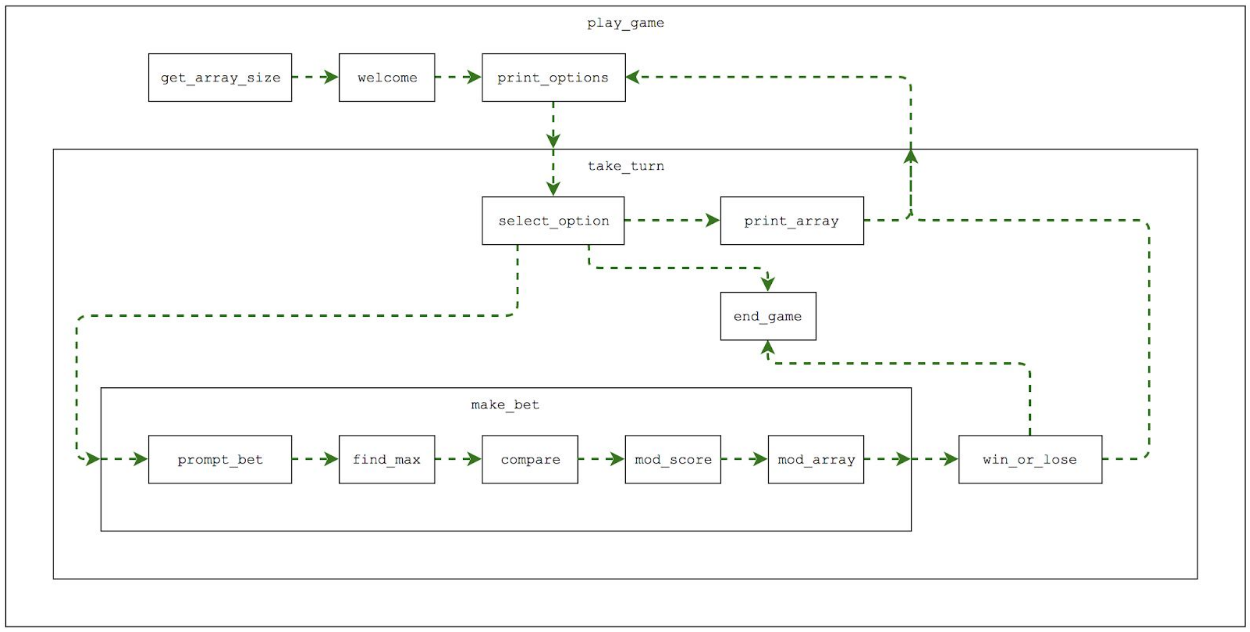

Figure: Program Flow

1. Subroutine play_game is called
2. Size of the array is determined
3. Welcome message is printed
4. Gameplay options are printed
5. Player types option
6. Subroutine take_turn is called
7. Depending on the option, one of the three subroutines is called (make_bet, print_array or end_game)
    a. make_bet
        i. Player enters bet amount and index guess
        ii. Subroutine find_max is called; the maximum value from the array is determined, and the the index of the max value is output
        iii. User guess and correct index for max value is compared
        iv. Score is modified according to the result
        v. If guess was correct, array is modified so max value is changed to -1
    b. print_array
    c. end_game
8. Repeat from step 4.

Subroutine Arguments

The following diagram describes how the subroutines interact with each other. Note that some subroutines are required, and others are optional. In addition, some program arguments are also optional.

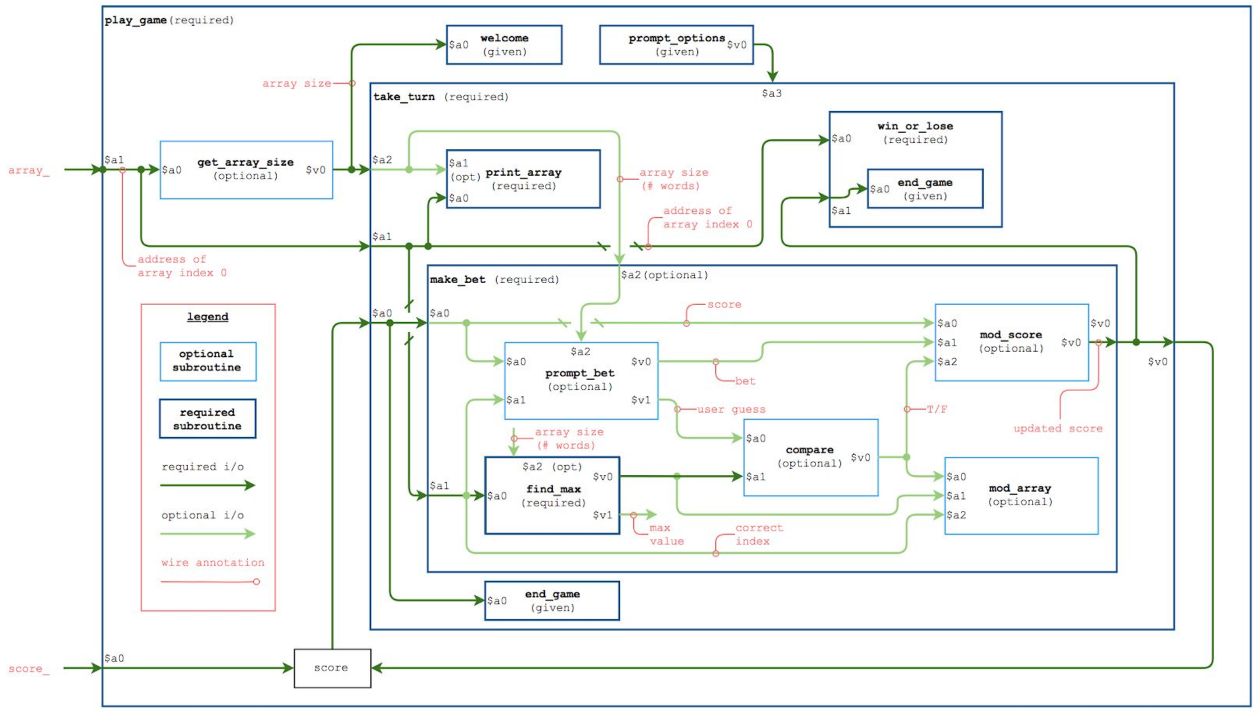

Figure: Subroutine I/O

The Stack

In this lab, you will use the program stack to handle the preservation of certain register values at the start of a subroutine so that they can be restored at the end of a subroutine. The stack should be used to preserve the jump and link return address, $ra, so you can navigate out of nested subroutines. In addition, the **values in registers $s0 - $s7 must be preserved across subroutine calls**.

Subroutines

Lab5.asm will contain the subroutines to execute the game. If you plan on using any of the saved registers, $s0 - $s7, you must save these registers appropriately on the stack (push at the beginning of your subroutine, then pop at the end of the subroutine).

You must use the JAL instruction to enter subroutines. Each subroutine will be tested individually; **your subroutines must function in isolation**.

Nested Subroutines

The required subroutines are nested as shown:

play_game

- welcome
- prompt_options
- take_turn

    - make_bet
        - find_max
    - print_array
    - win_or_lose
        - end_game
    - end_game

The subroutines *welcome*, *prompt_options*, and *take_turn* are called from *play_game*. The subroutines *make_bet*, *print_array*, *win_or_lose*, and *end_game* are called from *take_turn*. The subroutine *find_max* is called from *make_bet* and *end_game* can also be called from *win_or_lose*.

In order to properly execute these nested subroutines, you must use the stack to handle the return address register, $ra.

Required Subroutines

```
#--------------------------------------------------------------------
# play_game
#
# This is the highest level subroutine.
#
# arguments:  $a0 - starting score
#             $a1 - address of array index 0
#
# return:     n/a
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# welcome (given)
#
# Prints welcome message indicating valid indices.
# Do not modify this subroutine.
#
# arguments:  $a0 - array size in words
#
# return:     n/a
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# prompt_options (given)
#
# Prints user options to screen.
# Do not modify this subroutine. No error handling is required.
#
# return:     $v0 - user selection
#--------------------------------------------------------------------
```

```
#--------------------------------------------------------------------
# take_turn
#
# All actions taken in one turn are executed from take_turn.
#
# This subroutine calls one of following sub-routines based on the
# player's selection:
#
# 1. make_bet
# 2. print_array
# 3. end_game
#
# After the appropriate option is executed, this subroutine will also
# check for conditions that will lead to winning or losing the game
# with the nested subroutine win_or_lose.
#
# arguments:  $a0 - current score
#             $a1 - address of array index 0
#             $a2 - size of array (this argument is optional)
#             $a3 - user selection from prompt_options
#
# return:     $v0 - updated score
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# make_bet
#
# Called from take_turn.
#
# Performs the following tasks:
#
# 1. Player is prompted for their bet along with their index guess.
# 2. Max value in array and index of max value is determined.
#    (find_max subroutine is called)
# 3. Player guess is compared to correct index.
# 4. Score is modified
# 5. If player guesses correctly, max value in array is either:
#    --> no extra credit: replaced by -1
#    --> extra credit:    removed from array
#
# arguments:  $a0 - address of first element in array
#             $a1 - current score of user
#
# return:     $v0 - updated score
#--------------------------------------------------------------------
```

```
#--------------------------------------------------------------------
# find_max
#
# Finds max element in array, returns index of the max value.
# Called from make_bet.
#
# arguments:  $a0 - array
#
# returns:    $v0 - index of the maximum element in the array
#             $v1 - value of the maximum element in the array
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# win_or_lose
#
# After turn is taken, checks to see if win or lose conditions
# have been met
#
# arguments:  $a0 - address of the first element in array
#             $a1 - updated score
#
# return:     n/a
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# print_array
#
# Print the array to the screen. Called from take_turn.
#
# arguments:  $a0 - address of the first element in array
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# end_game (given)
#
# Exits the game. Invoked by user selection or if the player wins or
# loses.
#
# arguments:  $a0 - current score
#
# returns:    n/a
#--------------------------------------------------------------------
```

Optional Subroutines

These are suggested nested subroutines that you might want to implement in your program. You will not be graded on the functionality of these subroutines and it is not necessary to follow this format.

```
#--------------------------------------------------------------------
# get_array_size (optional)
#
# Determines number of 1-word elements in array.
#
# argument:   $a0 - address of array index 0
#
# returns:    $v0 - number of 1-word elements in array
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# prompt_bet (optional)
#
# Prompts user for bet amount and index guess. Called from make_bet.
#
# arguments:  $a0 - current score
#             $a1 - address of array index 0
#             $a2 - array size in words
#
# returns:    $v0 - user bet
#             $v1 - user index guess
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# compare (optional)
#
# Compares user guess with index of largest element in array. Called
# from make_bet.
#
# arguments:  $a0 - player index guess
#             $a1 - index of the maximum element in the array
#
# return:     $v0 - 1 = correct guess, 0 = incorrect guess
#--------------------------------------------------------------------

#--------------------------------------------------------------------
# mod_score (optional)
#
# Modifies score based on outcome of comparison between user guess
# correct answer. Returns score += bet for correct guess. Returns
# score -= bet for incorrect guess. Called from make_bet.
#
# arguments:  $a0 - current score
#             $a1 - player’s bet
#             $a2 - boolean value from comparison
#
# return:     $v0 - updated score
```

```
#------------------------------------------------------------------------

#------------------------------------------------------------------------
# mod_array (optional)
#
# Replaces largest element in array with -1 if player guessed correctly.
# Called from make_bet.
#
# If extra credit implemented, the largest element in the array is
# removed and array shrinks by 1 element. Index of largest element
# is replaced by another element in the array.
#
# arguments:  $a0 - address of array index 0
#             $a1 - index of the maximum element in the array
#
# return:     n/a
#------------------------------------------------------------------------
```

Extra Credit

When the user guesses the correct max element in the array, that element is removed from the array instead of replaced with -1. The array decreases in size shifting the elements at a larger index down to fill in the hole created from the element removal.

If the extra credit is implemented, win_or_lose will check if the array is empty.

Example

```
Given array:      [23 100000 42 14 33 8 11]

Player guess:     index 1

Original mod:     [23 -1 42 14 33 8 11]

Extra credit mod: [23 11 42 14 33 8] (one possibility)

Extra credit mod: [23 42 14 33 8 11] (another possibility)
```

Testing

Your program must work properly with the test program provided (on Canvas) lab5_test_19sp_cmpe012.asm.

For grading, each subroutine will be tested individually; **your subroutines must function in isolation**.

Files

Lab5.asm

This file contains your implementation of all subroutines. Follow the code documentation guidelines here. You are not required to write pseudocode for this lab.

README.txt

This file must be a plain text (.txt) file. It should contain your first and last name (as it appears on Canvas) and your CruzID. Your answers to the questions should total at least 150 words. Your README should adhere to the following template. For full credit, include the word *slug* somewhere in file. Posts to Piazza about this statement will result in failure of this lab. Note that there are blank lines before each question (these might not be selected if you copy-paste from the PDF instruction).

```
------------------------
Lab 5: A Gambling Game
CMPE 012 Spring 2019

Last Name, First Name
CruzID
-------------------------

What was your design approach?
Write the answer here.

What did you learn in this lab?
Write the answer here.

Did you encounter any issues? Were there parts of this lab you found
enjoyable?
Write the answer here.

How would you redesign this lab to make it better?
Write the answer here.

What resources did you use to complete this lab?
Write the answer here.

Did you work with anyone on the lab? Describe the level of collaboration.
Write the answer here.
```

Grading Criteria

This is a rough outline of the testing criteria. This is subject to change.

Overall

all subroutines work correctly with test code

correct final score

Subroutines

Note: credit will only be given for this section if Lab5.asm assembles on its own, and with test file

```
play_game
      gameplay executed properly

take_turn
      invokes make_bet if the user chose 1
      invokes print_array if the user chose 2
      invokes end_game if the user chose 3

make_bet
      invokes find_max
      updates score properly

find_max
      correctly returns the max index of array element
      correctly returns the max value of array element

print_array
      correctly prints array to screen

win_or_lose
      identifies win condition
      identifies lose condition

stack usage
      manages stack to save $s register values if using them

extra credit: mod_array
      shrinks array properly
```

Documentation / Style (8 pt)

```
6 pt  style and documentation
      1 pt comments on register usage (for each subroutine)
      1 pt useful and sufficient comments
      1 pt labels, instructions, operands, comments lined up in columns
      2 pt README contains at least 150 words (not including questions and header)
      total with complete thoughts
      1 pt complete headers for code and README
```

```
Note: program header must include name, CruzID, date, lab name, course name,
quarter, school, program description; README must include name, CruzID, lab
name, course name
```

Appendix

Sample Gameplays

Typical Playthrough

The extra credit was not implemented in this sample playthrough.

```
------------------------------
WELCOME
------------------------------

In this game, you will guess the index of the maximum value in an array. Valid
indices for this array are 0 - 1.

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

1

------------------------------
MAKE A BET

You currently have 100 points.
How many points would you like to bet? 150

Sorry, your bet exceeds your current worth.

You currently have 100 points.
How many points would you like to bet? 50

Valid indices for this array are 0 - 1.
Which index do you believe contains the maximum value? 1

Your guess is incorrect! The maximum value is not in index 1.

You lost 50 points.

------------------------------
CURRENT SCORE

50 pts

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet.
```

```
2 - Cheat! Show me the array
3 - Quit before I lose everything

2

------------------------------
CHEATER!

0: 44
1: 21

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

1

------------------------------
MAKE A BET

You currently have 50 points.
How many points would you like to bet? 20

Valid indices for this array are 0 - 1.
Which index do you believe contains the maximum value? 0

Score! Index 0 has the maximum value in the array.

You earned 20 points!

This value has been removed from the array.

------------------------------
CURRENT SCORE

70 pts

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

2

------------------------------
CHEATER!
```

```
0: -1
1: 21

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

3

------------------------------ GAME OVER ------------------------------

-- program is finished running --
```

Win Condition

```
------------------------------
WELCOME
------------------------------

In this game, you will guess the index of the maximum value in an array. Valid
indices for this array are 0 - 0.

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

1

------------------------------
MAKE A BET

You currently have 100 points.
How many points would you like to bet? 100

Valid indices for this array are 0 - 0.
Which index do you believe contains the maximum value? 0

Score! Index 0 has the maximum value in the array.

You earned 100 points!

This value has been removed from the array.

------------------------------
CURRENT SCORE
```

```
200 pts

------------------------------
YOU'VE WON! HOORAY! :D

------------------------------ GAME OVER ------------------------------

-- program is finished running --
```

Lose Condition

```
------------------------------
WELCOME
------------------------------

In this game, you will guess the index of the maximum value in an array. Valid
indices for this array are 0 - 64.

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

1

------------------------------
MAKE A BET

You currently have 100 points.
How many points would you like to bet? 100

Valid indices for this array are 0 - 64.
Which index do you believe contains the maximum value? 0

Your guess is incorrect! The maximum value is not in index 0.

You lost 100 points.

------------------------------
CURRENT SCORE

0 pts

------------------------------
YOU'VE LOST! D:

------------------------------ GAME OVER ------------------------------

-- program is finished running --
```

Extra Credit Implementation

```
------------------------------
WELCOME
------------------------------

In this game, you will guess the index of the maximum value in an array. Valid
indices for this array are 0 - 2.

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

2

------------------------------
CHEATER!

0: 2019
1: 12
2: 42

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

1

------------------------------
MAKE A BET

You currently have 100 points.
How many points would you like to bet? 100

Valid indices for this array are 0 - 2.
Which index do you believe contains the maximum value? 0

Score! Index 0 has the maximum value in the array.

You earned 100 points!

This value has been removed from the array.

------------------------------
CURRENT SCORE
```

```
200 pts

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

2

------------------------------
CHEATER!

0: 42
1: 12

------------------------------
What would you like to do? Select a number 1 - 3

1 - Make a bet
2 - Cheat! Show me the array
3 - Quit before I lose everything

3

------------------------------ GAME OVER ------------------------------

-- program is finished running --
```